APPROXIMATE POLYTREE LEARNING WITH SUBMODULAR LOCAL SCORES

Anonymous authors
Paper under double-blind review

ABSTRACT

Polytrees are a special case of Bayesian networks that seek to capture the conditional dependencies between variables as directed graphs whose underlying undirected graphs are acyclic. In the score-based approach to learning the structure of a polytree, we assign a score for each possible set of parents of each node to measure its quality, with the goal being to maximize the sum of the scores. As there is a variety of possible score functions, it is common to treat the scores as arbitrary blackbox functions, making structure learning NP-hard. At the other extreme, there has been research where additive scores have been assumed, which imposes a lot of structure on the score function but is an unrealistically strong assumption. We seek a middle ground and study the problem for submodular functions, to which the scores are empirically relatively close. We prove that the problem remains NP-hard for submodular scoring functions but also design a local-search algorithm that is guaranteed to find a structure whose score is within a factor of 2 from the optimal one, and if the maximum in-degree of the polytree is bounded, then a single iteration of the algorithm takes polynomial time. Moreover, we show empirically, that the number of iterations of our algorithm is small. Further, we show that this result can be generalized to functions that are close to being submodular without significantly worsening the approximation guarantee.

1 INTRODUCTION

Polytrees (PT) are probabilistic graphical models that attempt to capture the relationships between a set of variables as a tree that encodes the conditional dependencies. They are a special case of *Bayesian networks* (Koller & Friedman, 2009) that are made appealing by their interpretability and efficient inference (Pearl, 1989) and are used in image-segmentation for microscopy data (Fehri et al., 2019). One of the prevalent approaches to learning the structure of a polytree is the score-based approach, where each node is associated with a *local score function* that assigns each candidate set of parents a *local score* that measures how well it fits the data and the background assumptions (Cooper & Herskovits, 1992). The score of a polytree is then the sum of the local scores, and one looks for a structure that maximizes this score. However, similarly to general Bayesian networks (Chickering, 1995), learning the optimal structure is NP-hard (Dasgupta, 1999) and the best algorithms for the problem use exponential time (Grüttemeier et al., 2021a; Harviainen et al., 2026) and are unlikely to see significant improvements supposing the set cover conjecture from fine-grained complexity (Harviainen et al., 2026). Roughly speaking, the hardness follows from the local scores being treated as an arbitrary black box, which then allows to adversarially encode NP-hard problems as instances of polytree learning. Even if the local scores from the hardness proof are attainable for some specifically chosen score function, this is not necessarily possible for other common score functions like BDeu, BGe, or BIC (see, e.g., the survey of Kitson et al. (2023)).

We study whether *submodular* local score functions provide enough structure to the problem to make it more feasible. Roughly speaking, submodularity corresponds to us getting diminishing returns on the information gained by conditioning on a variable if we also condition on other variables. Intuitively, this is the case with correlated variables, which one often expects from real-world systems, albeit this is not always true (consider, e.g., xor-like variable relationships where knowing one variable gives no information about the output if the other is unknown). Submodularity is a relatively weak assumption in contrast to *additive* scores (Ganian & Korchemna, 2026; Harviainen et al., 2026; Scanagatta et al., 2016), whose relevance is mostly theoretical due to local scores computed from real

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

data not typically exhibiting such a strong structure—for example, the heuristic that inspired the study of additive scores has a logarithmic correction term (Scanagatta et al., 2016). With a small-scale empirical study, we evaluate how close the local scores are to being submodular under varying distance measures like the *submodularity ratio* (Das & Kempe, 2018) and α*-curvature* (Sviridenko et al., 2017), *marginal gain violation*, and the *set-based violation*. We observe that the scores tend to be relatively close to being submodular on the last metric, suggesting that algorithms assuming submodular scores may have potential in practice.

Even though we prove that structure learning remains NP-hard even for submodular local scores, we are also able to design an algorithm, for submodular local scores, that computes a 2-approximation of the optimal structure, that is, a polytree whose score is at least half of the optimal one. At a high level, the algorithm is based on local search and at each iteration picks a new parent set for some node such that we remove a subset of the existing arcs to maintain acyclicity while still improving the score by the addition of the new parents. Our approach for proving that the resulting graph approximates the optimal polytree well once the algorithm reaches a local optimum relies on utilizing the *generalized Rota-exchange property* (Greene & Magnanti, 1975; Lee et al., 2010; Schrijver et al., 2003) from the matroid literature to construct a mapping between the parent sets of the optimal polytree and subsets of arcs of the approximate polytree. The proof generalizes for scores that are close to being submodular with approximation guarantees that depend on the properties of the instances, and for 5 out of 8 benchmarks the attained guarantees outperform previous work (Harviainen et al., 2026).

In contrast to the more adaptive nature of our local search algorithm, the greedy algorithms of Harviainen et al. (2026) only add parent sets that preserve acyclicity, and they achieve approximation factors 2 and $k + 1$ for additive scores and arbitrary scores, respectively, with an in-degree bound k for the polytree. In other words, by allowing the greedy algorithm to "revert" some changes, we can obtain a constant factor approximation for submodular scoring functions. Limiting the in-degrees of the nodes is a traditional assumption in both theory (Bhattacharyya et al., 2022; Choo et al., 2024) and practice (Bartlett & Cussens, 2017; Cussens, 2020; Scanagatta et al., 2015) because it, for example, bounds the number of candidate parent sets to be polynomial in the number of nodes and decreases the sizes of the conditional probability tables. We also make this assumption, since otherwise the running time of a single iteration of our algorithm would be exponential in the number of nodes as it has to go through all candidate parent sets to see whether it has reached a local optimum. With a constant in-degree bound, the running time of each iteration becomes polynomial. The complexity increases exponentially in the in-degree bound, but one commonly chooses small bounds (Bartlett & Cussens, 2017; Scanagatta et al., 2015), keeping the running time feasible. We suspect that one can implement various optimization techniques to improve the running time of the algorithm in practice by, for example, ignoring some parent sets that cannot improve the score, but such algorithmic engineering is outside the scope of this work. Note that while imposing the in-degree bound simplifies many computations, learning the optimal polytree remains NP-hard (Dasgupta, 1999) except for the special case of $k = 1$ as shown by Edmonds (1967), and for arbitrary scores the approximation factor $k + 1$ is almost tight (Harviainen et al., 2026).

Finally, the running time depends also on the number of iterations the local-search algorithm takes to reach a local optimum. For local-search algorithms, proving an upper bound for the number of iterations is usually tedious and unhelpful, since for many problems the number of iterations is typically low for most practical instances except in some pathological cases (Etscheid & Röglin, 2017; Hartigan & Wong, 1979; Vattani, 2011). We create a proof-of-concept implementation of our algorithm to assess whether this is also the case for it, and on the tested instances the number of iterations is proportional to the number of variables.

Related work. Dasgupta (1999) studied the closely related problem of minimizing the sum of conditional entropies over the nodes; their approximation guarantees, however, do not transfer to the score-based setting we study in this work. Approximation algorithms are also studied for Bayesian networks (Kundu et al., 2024a; Ziegler, 2008). These works study general scoring functions without further assumptions such as submodularity. Different structural constraints to learn polytrees have been investigated especially through the lens of parameterized complexity (Gaspers et al., 2015; Ganian & Korchemna, 2021; Grüttemeier et al., 2021a). Moreover, the parameterized complexity of the more general Bayesian network structure learning is also well-studied (Grüttemeier & Komusiewicz, 2022; Ordyniak & Szeider, 2013; Kundu et al., 2024b; Grüttemeier et al., 2021b). More generally, submodularity is a common tool in approximation algorithms in ML (Bach, 2013; Wei et al., 2015).

2 PRELIMINARIES

Proofs marked by ★ are deferred to the appendix. Let D be a directed acyclic graph (DAG) over nodes $N = V(D)$ with a set of arcs $A = A(D)$. We set $n \coloneqq |N|$. The undirected graph obtained by removing the orientations of the edges of D is the *skeleton* of D. A DAG whose skeleton is a forest is a *polytree*. We may also call an arc set A a polytree if its skeleton is acyclic. The set of *parents* of a node $v \in N$ is denoted by D_v.

In the score-based approach to learning polytrees, we are given access to a *local score function* $s_v \colon 2^{N \setminus \{v\}} \to \mathbb{R}$ for each node $v \in N$ that assigns a value for each possible parent set of v. The *score of a DAG* D is the sum of the individual local scores, $s(D) \coloneqq \sum_{v \in N} s_v(D_v)$. If an arc set A induces a polytree on a node set N, we abbreviate the score of the corresponding DAG by $s(A) \coloneqq s((N, A))$. Moreover, if P is a parent set of a node v, we write $s(P) \coloneqq s_v(P)$. Also, we identify a parent set P of v with the corresponding set of incoming arcs $\{(u, v) \colon u \in P\}$.

To obtain an efficient encoding one often considers *non-zero encoding* of the input. There, the input contains a family $\mathcal{F}_v$ of potential parent sets for all $v \in N$ and their associated local scores $s_v(S)$ for all $S \in \mathcal{F}_v$. All other local scores are assumed to be $-\infty$. Moreover, we assume that the local scores are normalized in such a way that $s_v(\emptyset) = 0$ for all nodes v by shifting all local scores of each node by an appropriate amount, similarly to Kundu et al. (Kundu et al., 2024a) and Ziegler (Ziegler, 2008). This is beneficial since now the empty graph has score 0 and thus so does any optimal polytree. The general problem now reads as follows.

Polytree Learning (PT)
Instance: A node set N, and local score functions s_v for $v \in N$.
Task: Compute a polytree $D = (N, A)$ with maximum score.

If we are guaranteed that the largest parent set in the instance with a specified score has size k, we add a subscript '$\leq$' to the problem name (also later).

A local score function s_v is *additive* if we have that $s_v(S) = \sum_{u \in S} s_v(\{u\})$. Thus, if we deal with additive scores, we assume the input to contain the $\mathcal{O}(n^2)$ values $s_v(\{u\})$ for all distinct $v, u \in N$. The score function is *additive* if all local scores are additive.

SUBMODULAR (SCORING) FUNCTIONS. Let U be a universe and let $f : 2^U \to \mathbb{R}$ be a set function. For disjoint sets $A, B \subseteq U$, we write $\mathsf{mG}(A \mid B) \coloneqq f(A \cup B) - f(B)$ for the *marginal gain* of adding A on top of B. For singletons $u \in U \setminus B$ we simplify notation to $\mathsf{mG}(u \mid B) = f(B \cup \{u\}) - f(B)$. Function f is called *submodular* if for all $A, B \subseteq U$ the property $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$ holds, or equivalently, if for all $A \subseteq B \subseteq U$ and $X \subseteq U \setminus B$ we have $\mathsf{mG}(X \mid A) \geq \mathsf{mG}(X \mid B)$. Function f is *monotone increasing* if for each $A \subseteq B \subseteq U$ we have $f(B) \geq f(A)$, or equivalently, if for all $A \subseteq U$ and $u \in U \setminus A$ we have $\mathsf{mG}(u \mid A) \geq 0$.

In this work, we consider polytree learning with monotone and submodular scoring functions, that is, we assume that each scoring function s_v is monotone and submodular. Here, we do *not* assume the above-described non-zero representation of the parent sets. Instead we assume that the scoring functions are given via the *maximal parent sets $\mathcal{F}_v$ of any node* $v \in N$.

The input consists, for each $v \in N$, of an antichain $\mathcal{F}_v$ of *maximal parent sets* of size at most k together with the score of every subset P of every $D_v \in \mathcal{F}_v$; we call these parent sets *stored*. We let $\mathcal{F} \coloneqq \bigcup_{v \in N} \mathcal{F}_v$. Also, for every other set $Q \subseteq N \setminus \{v\}$ we define $s_v(Q) \coloneqq \max\{s_v(P) \colon P \subseteq Q,\ P \text{ stored}\}$. Thus, if the largest maximal parent set size k is a constant, then the total number of stored parent sets is $2^k \cdot |\mathcal{F}|$. This is polynomial in the input size since $|\mathcal{F}|$ is part of the input. If k is not a constant, we use an efficient encoding via *minimal marginal gains*; their description is deferred to the appendix since we only exploit this encoding in the proof of Theorem 3.3 (in the appendix). By $|\mathcal{I}|$ we denote the length of the encoding of the input, which is $|\mathcal{F}|$ up to polynomial factors in n.

We study the following problem.

Submodular Polytree Learning (PT^{sm})
Instance: A node set N, and monotone submodular local score functions s_v for $v \in N$.
Task: Compute a polytree $D = (N, A)$ with maximum score.

By $\text{PT}^{\text{sm}}_{\leq}$ we denote the variant of PT^{sm} where we impose a maximum parent set size constraint of k.

DEFINITION OF FEEDBACK EDGE SET $\mathsf{fes}(D_v)$. Let A be a polytree on a node set N and let $(D_v)_{v\in N}$ be the parent sets of A. For an acyclic arc set Z over $N \times N$, the *feedback edge set* $\mathsf{fes}(Z)$ is the set of arcs of $A \setminus Z$ of minimal total score such that $(A \setminus \mathsf{fes}(Z)) \cup Z$ is a polytree. Formally, for fixed $Z \subseteq N \times N$ let $X \subseteq A$ be any arc set. If $(A \setminus X) \cup Z$ is a polytree, we set $g(X) := s(A) - s(A \setminus X)$, and otherwise we set $g(X) := \infty$. Now, $\mathsf{fes}(Z)$ be any set $X \subseteq A$ which minimizes $g(X)$. Note that since A is a polytree, $\mathsf{fes}(Z)$ always exists such that $g(\mathsf{fes}(Z)) \leq s(A)$. Moreover, note that $|\mathsf{fes}(Z)| \leq |Z|$ since s is monotone, since monotonicity implies that no subset of arcs of a parent set can be removed without decreasing the score.

APPROXIMATION ALGORITHMS AND PARAMETERIZED ALGORITHMS. For a constant $c > 1$, an algorithm $\mathcal{A}$ for PT (and its variants) is a *c-approximation algorithm* if it always outputs a polytree D whose score is within a factor c from the optimal polytree D^{opt}, that is, $f(D) \leq f(D^{\text{opt}}) \leq c \cdot f(D)$.

We study the complexities of these problems from the viewpoint of parameterized complexity (Cygan et al., 2015; Downey & Fellows, 1995b;a). A problem is *fixed-parameter tractable* (FPT) with respect to a parameter p if any instance $\mathcal{I}$ of the problem is solvable in time $\mathcal{O}(f(p) \cdot \mathsf{poly}\,|\mathcal{I}|)$ for some computable function f. Running time lower bounds can be proven conditional to the exponential time hypothesis (ETH) (Impagliazzo & Paturi, 2001; Impagliazzo et al., 2001), which states that 3-SAT on n-variable formulas cannot be solved in $2^{o(n)}$ time. Essentially, one then shows that a fast algorithm to a problem would violate ETH through a reduction from 3-SAT to the problem.

3 A 2-APPROXIMATION FOR SUBMODULAR SCORING FUNCTIONS

Now, we provide an algorithm `LS-PT` which yields a 2-approximation for polytree learning if the scores are monotone and submodular. The basic idea of `LS-PT` is to use local search. More specifically, each iteration aims to improve the score of a given polytree T by adding a new parent set P_v to T, while simultaneously removing the existing parent set of v in T and some further arcs of T to ensure acyclicity. A straightforward greedy algorithm that iteratively adds a parent set with highest score that preserves acyclicity can yield bad approximation factors, see Figure 1 (in the appendix). Hence, the ability to remove arcs is essential to reach the approximation guarantee of 2. Later, we generalize the theorem to local score functions that are close to being submodular.

Theorem 3.1. $\text{PT}^{\text{sm}}_{\leq}$ *and* PT^{sm} *admit a 2-approximation algorithm.*

3.1 DESCRIPTION OF OUR ALGORITHM `LS-PT`

We now present our local-search algorithm `LS-PT`; see also Algorithm 1. Initially in Line 1, we compute all *reasonable* parent sets. A parent set P_v for a node v is *reasonable* if for every $Q_v \subset P_v$, we have $s(Q_v) < s(P_v)$. Clearly, if a polytree A contains a non-reasonable parent set P_v, that is there exists a parent set $Q_v \subset P_v$ with $s(Q_v) \geq s(P_v)$, then, one can replace P_v by Q_v and obtains another polytree with larger or equal score. Thus, it is sufficient that `LS-PT` only adds reasonable parent sets. Note that all reasonable parent sets can be identified via brute force.

We start with an empty polytree A (Line 2). As described in Figure 1 (in the appendix), to avoid getting stuck in bad local optima, our algorithm not only adds new arcs to A but is also allowed to remove some of the existing arcs to ensure acyclicity. More precisely, in each iteration we do the following (Line 4): For each possible parent set D_v of some node $v \in N$, we try adding D_v to A (Lines 8 and 10). More precisely, D_v is a subset of a maximal parent set in $\mathcal{F}_v$. To do so, we also remove the existing parent set D^v_A of v in A. Since this might not be sufficient to ensure that the resulting arc set induces a polytree, we also remove any arc set Z such that $(A \setminus (D^v_A \cup Z)) \cup D_v$ is a polytree (Line 11). If this yields a polytree with higher score, we stop this iteration (Line 12). Note that since `LS-PT` only adds reasonable parent sets, we always have $|\mathsf{fes}(Z)| \leq |Z|$. Note that we get this property from the monotonicity of the score functions as well, but we use this formulation to later argue that the assumption of f being monotone is not necessary. We do this until we reach a local optimum, that is, no more improving iteration can be found (Line 3).

Algorithm 1 Algorithm `LS-PT` to compute a polytree. The algorithm is started by calling `ComputePolytree`.

Function `ComputePolytree`
Input: A node set N and a family of parent sets $\mathcal{F}$.
Output: A polytree with arc set A.
For each $v \in N$: compute $\mathcal{F}_v^\star$ from $\mathcal{F}_v$
$A := \emptyset$ `// initialize the empty polytree`
while `True` **do**
  $A' :=$ `OneStepImprovement` (A)
  if $A' = A$ **then break**
  $A := A'$
return A

Function `OneStepImprovement` (A)
Input: A polytree on N with arc set A.
Output: Another polytree A_v such that (a) $s(A_v) > s(A)$ (b) A_v consists of one new parent set D_v of some node v compared to A, and (c) the existing parent set D_A^v of v as well as a cheapest arc set $\mathsf{fes}(D_v)$ are removed from A such that A_v is a polytree.
foreach $v \in N$ **do**
  $D_A^v :=$ parent set of v in A
  foreach $D_v \in \mathcal{F}_v^\star$ **do**
    Find a cheapest arc set $\mathsf{fes}(D_v) \subseteq A \setminus D_A^v$ such that $A_v := (A \setminus (D_A^v \cup \mathsf{fes}(D_v))) \cup D_v$ is a polytree
    if $s(A_v) > s(A)$ **then return** A_v
return A

3.2 Sketch of Proof of Theorem 3.1

We now show that the score of the polytree with arc set A returned by Algorithm 1 is a 2-approximation of score of an optimal polytree S if the score function is monotone and submodular.

Overall strategy for showing the approximation guarantee. Let A be an approximate (locally optimal) solution computed by `LS-PT` (see Algorithm 1) and let S be an optimal polytree. We show the 2-approximation as follows: Since A is locally optimal, it is not possible to add any further parent set D_v, while also removing the existing parent set D_A^v of v in A and any arc set Z of A such that $A'_v := (A \setminus (D_A^v \cup Z)) \cup D_v$ is a polytree.

Note that since this applies to *every* Z, it in particular applies to the cheapest arc set $\mathsf{fes}(D_v)$ in terms of the resulting score, that is, among all Z for which $A_v := (A \setminus (D_A^v \cup \mathsf{fes}(D_v))) \cup D_v$ is a polytree, the arc set $\mathsf{fes}(D_v)$ maximizes the resulting score. In particular we have $s(A_v) \geq s(A'_v)$. Note that since `LS-PT` only adds reasonable parent sets, we always have $|\mathsf{fes}(Z)| \leq |Z|$ as no subset of edges of any parent set can be removed without decreasing the resulting score.

Since this property applies to any parent set D_v, it in particular applies to any parent set D_S^v of the optimal solution S. The local optimality of A implies that $s(D_S^v) \leq s(D_A^v) + \mathsf{mG}(\mathsf{fes}(D_S^v))$, where $\mathsf{mG}(\mathsf{fes}(D_S^v))$ is the *marginal gain* of the arcs of $\mathsf{fes}(D_S^v)$. Afterwards, by summing the above inequality over all nodes $v \in N$, we obtain that the left side is equal to the score $s(S)$ of the optimal solution. We decompose the right side into the sum over all $s(D_A^v)$, which we show to be equal to the score $s(A)$ of the approximate solution, and the sum over all $\mathsf{mG}(\mathsf{fes}(D_S^v))$, which we show to be bounded by $s(A)$ as well. This then yields the desired 2-approximation.

The most technical part is showing that the sum of $\mathsf{mG}(\mathsf{fes}(D_S^v))$ over all parent sets of the optimal solution S is upper bounded by $s(A)$. One issue is that an arc e of the approximate solution A might be contained in many different sets $\mathsf{fes}(D_S^v)$. In order to still obtain an upper bound we extend both polytrees A and S by adding artificial dummy arcs such that their underlying graphs are spanning trees on the entire node set N. Then, we can interpret them as bases of so-called *graphical matroids* and apply a generalized Rota-exchange property (Greene & Magnanti, 1975; Schrijver et al., 2003; Lee et al., 2010) to show the desired statement. Intuitively, this property corresponds to an *exchange mapping* f: We map each parent set D_S^v of the optimal polytree S to an arc set $f(D_S^v)$ of the

(approximate) polytree A of size (at most) $|D_S^v|$ such that a) no arc of A is hit twice and b) replacing the arcs of $f(D_S^v)$ in A by the arcs of D_S^v yields a polytree $A_v^\star$ again.

We need both properties: We need b) to argue that since $\mathsf{fes}(D_S^v)$ is the cheapest arc set we can remove such that A'_v is a polytree, the score of $A_v^\star$ cannot be higher than that of A_v. Property a) then allows us to avoid the above-mentioned issue that some arcs e of the approximate solution A might be contained in many different sets $\mathsf{fes}(D_S^v)$. We then can exploit the monotonicity and the submodularity of the scoring functions to compose the arcs of all sets $\mathsf{fes}(D_S^v)$ into the parent sets of the approximate polytree A. Hence, this allows us to bound the overall score of all $\mathsf{fes}(D_S^v)$ arcs by $s(A)$. We defer the formal proof and the definition of an exchange mapping f to the appendix. □

Limits of the approximation guarantee. The approximation guarantee of LS−PT (Algorithm 1) shown in Theorem 3.1 for $\mathrm{PT}^{\mathrm{sm}}_{\leq}$ and $\mathrm{PT}^{\mathrm{sm}}$ cannot be improved. For an example, we refer to Figure 3 (in the appendix). A discussion on whether LS−PT (Algorithm 1) also yields a better approximation guarantee than $k+1$ for polytree learning with general scoring functions is deferred to the appendix.

3.3 Running-Time Analysis

It remains to analyze the running time of LS−PT (Algorithm 1). For this, we need to bound (1) the number of iterations, (2) the pre-processing time required to compute all reasonable parent sets, and (3) the running time per iteration. For (1) the maximal number of iterations, we could not prove any bound. We, however, suspect that in the worst case an exponential number of iterations (with respect to $|N|$) could be necessary. More precisely, we assume that the corresponding problem is PLS-hard (Johnson et al., 1988). Verifying or disproving this is an important challenge of future work. These results, however, only concern the worst-case, and in practice PLS-hard problems often only require a polynomial number of iterations until a local optimum is reached (Arthur et al., 2011). In Section 4.2, our experiments demonstrate that the number of iterations is often proportional to the number of nodes for our problem. For (2) observe that for each subset $D_v \in \mathcal{F}_v$ of a maximal parent set, by inspecting every $D_v^\star \subseteq D_v$ we can check whether D_v is reasonable or not. This requires $\mathcal{O}(4^k \cdot |\mathcal{F}|)$ time. Next, we resolve (3), that is, we bound the running time per iteration. We now show that one iteration of LS−PT can be done in FPT-time with respect to k (single-exponential in k), that is, in polynomial time if k is a constant.

Theorem 3.2 (★). *Each iteration of LS−PT (Algorithm 1) can be done in $4^k \cdot |\mathcal{F}| \cdot \mathcal{O}(n)$ time, where $\mathcal{F}$ denotes the set of maximal parent sets and k the largest maximal parent set size.*

One immediate follow-up question from Theorem 3.2 is whether the constraint that the maximal parent set size k is a constant is necessary. In other words, the question is whether one iteration can be done in polynomial time independent of the value of k. We show that this is not possible. More precisely, we show that given a polytree T on a node set N, it is NP-hard to find a polytree $T^\star$ which has higher score than the initial polytree T such that $T^\star$ can be obtained by one iteration of LS−PT from T. In our reduction two properties will hold: (a) there is exactly one candidate parent set D_v which can be chosen in Line 10 of LS−PT such that it is possible to have $s(T^\star) > s(T)$, and (b) we have $s(T^\star) > s(T)$ if and only if arc set $\mathsf{fes}(D_v)$ is removed from T (Line 11 of Algorithm 1). In other words, requiring to remove *any* arc set Z such that $T^\star := (T \setminus (D_T^v \cup Z)) \cup D_v$ fulfills $s(T^\star) > s(T)$, where D_T^v is the current parent set of v, in T does *not* yield a better worst-case behavior of LS−PT. Hence, the following theorem (Theorem 3.3) implies that even if we are given a new parent set D_v which we aim to add to T, the determination of an optimal arc removal set $\mathsf{fes}(D_v)$ (Line 11 of Algorithm 1) is NP-hard. Note that (a) implies that $T^\star[V-v]$ is a subgraph of $T[V-v]$.

We now show that the following problem is NP-hard. Here, we exploit the efficient encoding of all parent sets described in Section 2 (see appendix): It is not sufficient to only store the maximal parent sets since the size of one parent set in the input is linear in the number of nodes. Hence, storing the scores of all subsets would not be polynomial in the input size. We show, however, that the size of the encoding of the scores via the minimal marginal gains is only polynomial in the number of nodes.

Best Improved Polytree

Instance: A polytree T over N, and monotone submodular local score function f_v for all $v \in N$.

Task: Find a polytree $T^\star$ obtainable by one iteration of LS−PT from T such that $s(T^\star) > s(T)$.

Theorem 3.3 (★). *Even if T has exactly one node with indegree at least two,* BEST IMPROVED POLYTREE *is NP-hard and cannot be solved in $2^{o(n)}$ time unless the ETH fails.*

We next show that both $\mathrm{PT}^{\mathrm{sm}}$ and $\mathrm{PT}^{\mathrm{sm}}_{\leq}$ are NP-hard. Here, we only show the result for $\mathrm{PT}^{\mathrm{sm}}_{\leq}$ since the result for $\mathrm{PT}^{\mathrm{sm}}$ is a direct consequence.

Theorem 3.4 (★). *For each $k \geq 4$, $\mathrm{PT}^{\mathrm{sm}}_{\leq}$ is NP-hard and cannot be solved in $2^{o(n)}$ time unless the ETH fails.*

3.4 GENERALIZATION TO NON-SUBMODULAR FUNCTIONS.

Our results generalize to functions which are not submodular. We will next introduce several metrics for how close a function is to being submodular and then describe the approximation guarantees we obtain for them. A more detailed description of these metrics is in the appendix. Also, the proofs are in the appendix and are adaptations of the proof of Theorem 3.1.

1. Das & Kempe (2018) defined the *submodularity ratio* γ of f over a universe U where $U' \subseteq U$ and $k \leq |U'|$ as $\gamma_{U',k} := \min_{\substack{S,T \subseteq U', \\ |T| \leq k, S \cap T = \emptyset}} \frac{\sum_{u \in T}(f(S \cup \{u\}) - f(S))}{f(S \cup T) - f(S)}$. In this work, we always set $U' = U$ and k denotes the maximal parent set size. Thus, we write γ_k instead. If we consider the variant with unbounded parent set size, we write γ.

2. The definition of α*-curvature* implies that for a monotone increasing set function f we have $(1 - \alpha)\,\mathsf{mG}(u \mid B) \leq \mathsf{mG}(u \mid A)$ for each $u \in U$ and each $A, B \subseteq U \setminus \{u\}$ (Sviridenko et al., 2017, Equation 7).

3. The *marginal gain violation (*mG*-violation)* ϵ_{mG} of a monotone non-decreasing set function f is the smallest number $\epsilon_{\mathsf{mG}} \geq 0$ such that $\mathsf{mG}(C \mid A) \leq (1 + \epsilon_{\mathsf{mG}}) \cdot \mathsf{mG}(C \mid B)$ for all $A, B, C \subseteq U$ such that $C \cap (A \cup B) = \emptyset$ and $B \subseteq A$. Moreover, the *element-wise marginal gain violation (element* mG*-violation)* $\epsilon^{ele}_{\mathsf{mG}}$ of a monotone non-decreasing set function f is the smallest number $\epsilon^{\mathrm{ele}}_{\mathsf{mG}} \geq 0$ such that $\mathsf{mG}(u \mid A) \leq (1 + \epsilon^{\mathrm{ele}}_{\mathsf{mG}}) \cdot \mathsf{mG}(u \mid B)$ for all $A, B \subseteq U$, $u \in U$ such that $\{u\} \cap (A \cup B) = \emptyset$ and $B \subseteq A$.

4. We define the *set-based violation* ϵ_{set} as the smallest $\epsilon_{\mathrm{set}} \geq 0$ such that $(1 + \epsilon_{\mathrm{set}}) \cdot (f(S) + f(T)) \geq f(S \cup T) + f(S \cap T)$ for all S, T. As in 3., we have $\epsilon_{\mathrm{set}} = 0$ if and only if f is submodular.

Based on the above definitions, we consider the following variants of PT(f is always also monotone):

1. $\mathrm{PT}^{(\alpha,\gamma)}$, with curvature $0 < \alpha < 1$ and submodularity ratio $0 < \gamma < 1$;
2. $\mathrm{PT}^{\mathsf{mG}}$, with marginal gain violation ϵ_{mG};
3. $\mathrm{PT}^{\mathsf{mG},\gamma}$, with element-wise marginal gain violation $\epsilon^{\mathrm{ele}}_{\mathsf{mG}}$ and submodularity ratio $0 < \gamma < 1$;
4. $\mathrm{PT}^{\mathrm{set}}$, with set-based violation ϵ_{set}.

By $\mathrm{PT}^{(\alpha,\gamma)}_{\leq}$, $\mathrm{PT}^{\mathsf{mG}}_{\leq}$, $\mathrm{PT}^{\mathsf{mG},\gamma}_{\leq}$, and $\mathrm{PT}^{\mathrm{set}}_{\leq}$ we denote the variants of $\mathrm{PT}^{(\alpha,\gamma)}$, $\mathrm{PT}^{\mathsf{mG}}$, $\mathrm{PT}^{\mathsf{mG},\gamma}$, and $\mathrm{PT}^{\mathrm{set}}$ where we impose a maximum parent set size constraint of k, respectively.

Theorem 3.5 (★). *1. $\mathrm{PT}^{(\alpha,\gamma)}_{\leq}$ and $\mathrm{PT}^{(\alpha,\gamma)}$ admit a $(1 + 1/(\gamma \cdot (1 - \alpha)))$-approx. algorithm.*
2. $\mathrm{PT}^{\mathsf{mG}}_{\leq}$ and $\mathrm{PT}^{\mathsf{mG}}$ admit a $(2 + \epsilon_{\mathsf{mG}})$-approximation algorithm.
3. $\mathrm{PT}^{\mathsf{mG},\gamma}_{\leq}$ and $\mathrm{PT}^{\mathsf{mG},\gamma}$ admit a $(1 + (1 + \epsilon^{ele}_{\mathsf{mG}})/\gamma)$-approximation algorithm.

Theorem 3.6 (★). $\mathrm{PT}^{\mathrm{set}}_{\leq}$ *admits a $\big(2 + 2k \cdot \epsilon_{set}/(1 + \epsilon_{set})\big)$-approximation algorithm. Recall that k is the maximum parent-set size.*

Note that unlike Items 1–3 in Theorem 3.5, the approximation guarantee in Theorem 3.6 *depends* on the maximum parent-set size k. The reason is that (cf. proof in the appendix) ϵ_{set} bounds the marginal-gain overshoot by $\epsilon_{\mathrm{set}} \cdot (f(S) + f(T))$ rather than by a multiple of the marginal gain itself, so the slack term arising in the proof of Theorem 3.5 does not telescope across the chain decomposition and accumulates a $\Theta(k)$ factor.

Table 1: *Top:* Submodularity-related quantities per dataset: maximum set-based violation ϵ_{set}, submodularity ratio γ, lower bounds $\epsilon^{\text{ele}}_{\mathsf{mG},k}$ and α_k on the element-wise marginal-gain violation $\epsilon^{\text{ele}}_{\mathsf{mG}}$ and curvature α, largest parent set size k after pruning, the set-based approximation factor on the (0-shifted, pruned) BDeu scores, and approximation factor $k+1$ of Harviainen et al. (2026) for arbitrary scores. The k-subscripted variants are restricted to the largest parent-set sizes after pruning. Values marked with * are lower bounds rather than exact; since a lower bound on ϵ_{set} yields a lower rather than an upper bound on the approximation factor, for mushroom we instead report the valid factors obtained in the limit $\epsilon_{\text{set}} \to \infty$. *Bottom:* Number of iterations of `LS-PT` (Algorithm 1) until a local optimum is reached; one iteration is one improving move. ***Steepest*** selects the move with the largest score gain, ***first-imp.*** draws a fresh uniformly random node order in every iteration and takes the best move at the first node that admits one. Values are over 25 random seeds for each dataset and rule. $|\mathcal{F}|$ is the number of stored parent sets after pruning.

	adult	car	chess-KRK	dna	mushroom	nursery	solar-flare	vote
ϵ_{set}	0.2935	0.5757	3.71	0.0984*	24.45*	1.23	0.1621	0.1616
$\epsilon_{\text{set},k}$	0.2935	0.5757	3.71	0.0984	24.45*	1.23	0.1621	0.1436
$\epsilon^{\text{ele}}_{\mathsf{mG},k}$	∞	∞	∞	∞	∞	∞	∞	∞
γ	0	0.3129	0.1292	0	0	0.4483	0	0
γ_k	0	0.3129	0.2121	0	0	0.4483	0	0.6705
α_k	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
k	3	4	3	4	10	4	3	3
$k+1$	4	5	**4**	5	**11**	**5**	4	4
$2+2k\,\frac{\epsilon_{\text{set},k}}{1+\epsilon_{\text{set},k}}$	**3.36**	**4.92**	6.73	**2.72**	22	6.41	**2.84**	**2.75**
n	14	7	7	61	22	9	7	17
$\lvert\mathcal{F}\rvert$	406	35	76	460	455336	61	82	480
iter., steepest	13	4	4	57	22	5	6	17
iter., first-imp.	15.8	4.8	5.8	59.1	31.2	6.5	8.2	18.7
range, first-imp.	13–18	4–6	4–9	57–62	23–39	5–10	7–10	15–21

4 Empirical Results

4.1 Distance to Submodularity and Implied Approximation Factors

We now report on the properties of score functions encountered by a standard approach to polytree learning (and more generally Bayesian network structure learning) on standard benchmark datasets and the approximation factors obtained by using our algorithms. In a nutshell, the curvature, submodularity ratio, and marginal-gains violations are large, making the functions far from submodular when measured in this way. On the positive side, they are often close to being submodular when measured by set-based violations and suitable parent-set sizes are often small, yielding approximation factors via Theorem 3.6 that improve on the state of the art (Harviainen et al., 2026, Theorem 4.1).

We proceeded as follows.[1] The eight datasets were selected as by Cheng & Greiner (1999) from the UCI machine learning repository (Kelly et al., 2007) and we analogously discretized them using WEKA's (Frank et al., 2016) implementation of the MDL method (Fayyad & Irani, 1993). We then computed BDeu scores (Buntine, 1991; Heckerman et al., 1995) using Gibby (Nikzad et al., 2025) with the pruning turned off in order to preserve the optimal solutions. Since in general there is an exponential number of potential parent-set scores to compute, we used binary search to find the largest upper bound on the parent set size such that Gibby finished computing all the scores within 10 min. This was the case for all datasets except for dna, where the largest parent-set size was 4 with in total 61 variables.

[1] The full, reproducible code pipeline is provided in the supplemental material.

We then pruned all unreasonable parent sets, that is, parent sets T for which there is a parent set $S \subsetneq T$ with larger score. This results in a monotone scoring function. Note that this change does not necessarily retain submodularity of the scores, but affects neither the optimal solution nor the approximation algorithm. The latter follows from the fact that the LS−PT ignores non-reasonable parent sets. At this point, we recorded for each dataset the largest (unpruned) parent-set size k. Note that without loss of generality every solution only includes sets of size at most k. Since BDeu scores are negative, we shift each variable's local score s_v by its minimum so that the resulting scores are nonnegative and $s_v(\emptyset) = 0$. This is in the same spirit as the score transformations used in shortest-path formulations of structure learning (Yuan & Malone, 2013) and prevalent in literature on approximation algorithms (Harviainen et al., 2026; Kundu et al., 2024a; Ziegler, 2008) and additive scores (Ganian & Korchemna, 2026) for Bayesian networks. Note that this step does not necessarily preserve approximation factors; we show below how it affects them in our datasets.

We then computed γ, ϵ_{set}, $\epsilon^{\text{ele}}_{\mathsf{mG}}$, and α of the resulting functions, see Table 1 for their values. Computing them usually involves enumerating an exponential number of (pairs of) subsets of variables. This is infeasible for some of the data sets and hence we computed bounds (marked by a star) obtainable within 10 min. In the proofs of Theorem 3.5 and Theorem 3.6 the properties of the score functions are only used for sets of size up to k. Hence, we also computed variants, subscripted by k, which take only sets of size at most k into account. For instance, for α_k we restrict in the definition $|A|, |B| \leq k$. Usually, they match with the unrestricted variants, but for instance the submodularity ratio for the vote dataset is drastically different.

As mentioned, the resulting score functions have extreme curvature, and they are far from being submodular when measured in submodularity ratio (range 0 to 1) and marginal-gains violation (range 0 to ∞). This is because in several cases, adding a variable to a set does not increase its score (yielding zero marginal gain after pruning), whereas a combination with a specific other variable will increase it (yielding positive marginal gain). This forces $\epsilon^{\text{ele}}_{\mathsf{mG}}$ to ∞, for instance. In contrast, when measured in terms of violation ϵ_{set} (range 0 to ∞) of the set-based submodularity inequality, several of the data sets are close to being submodular. Furthermore, applying the approximation guarantee from Theorem 3.6, we obtain guaranteed approximation factors that improve on the state-of-the-art of $k + 1$ (Harviainen et al., 2026, Theorem 4.1).

4.2 Number of Iterations until a Local Optimum

In Section 3.3 we bounded the cost of a single iteration of LS−PT but left the number of iterations open. We now measure these on the data from above. We start from the empty polytree and iteratively apply Algorithm 1, computing $\mathsf{fes}(D_v)$ exactly in every step, until the algorithm terminates in a local optimum. Algorithm 1 leaves open in which order improving moves are applied, which affects the number of iterations. We therefore ran two pivot rules: *steepest ascent*, which applies the move with the largest gain, and a *randomised first-improvement* rule, which draws a fresh uniformly random node order in each iteration and applies the best move at the first node that admits one.

Table 1 shows that the number of iterations is small and scales roughly linearly with the number n of nodes: it lies between $0.56\,n$ and $1.77\,n$ over all datasets and both pivot rules, and the largest count we observed was 62, on dna with $n = 61$. This supports the expectation that, even if the problem is PLS-hard, a polynomial—here essentially linear—number of iterations suffices in practice.

5 Conclusion

We initiated the study of polytree learning in the score-based approach if we assume the scoring functions to be submodular. We showed that a simple local-search based algorithm can achieve a 2-approximation of the optimal score. Moreover, one iteration of this algorithm can be done in polynomial time if the maximal parent set size is constant. We also performed a small scale experiment and observed that practical scoring functions are often close to being submodular. We also showed that our 2-approximation can be generalized to some scoring functions which are close to be submodular.

An immediate open question is whether the number of iterations of our local search algorithm is polynomial in the number of nodes or not. Moreover, it is open whether one can improve upon the approximation factor of 2. More general, it is interesting to study further models for scoring functions which occur in practical data.

AI USE STATEMENT

Original theoretical content from AI is limited to Claude Code contributing an initial version of Theorem 3.6 based on Theorem 3.5 and improving the running time of the local-search algorithm in Theorem 3.2 to $4^k \cdot |\mathcal{F}| \cdot \mathcal{O}(n)$ based on the initial FPT-time algorithm of running time $2^{6k} \cdot |\mathcal{F}| \cdot \mathcal{O}(\mathsf{poly}(|\mathcal{I}|))$ by the authors, where $\mathcal{F}$ denotes the set of maximal parent sets, k the largest maximal parent set size, and $|\mathcal{I}|$ the total input size. Both results were edited and rigorously checked by all the authors. Further, Claude Code was used to implement the algorithm for the experiment assessing the number of iterations the local search takes.

REFERENCES

David Arthur, Bodo Manthey, and Heiko Röglin. Smoothed analysis of the k-means method. *J. ACM*, 58(5):19:1–19:31, 2011.

Francis R. Bach. Learning with submodular functions: A convex optimization perspective. *Found. Trends Mach. Learn.*, 6(2-3):145–373, 2013.

Mark Bartlett and James Cussens. Integer linear programming for the Bayesian network structure learning problem. *Artif. Intell.*, 244:258–271, 2017.

Arnab Bhattacharyya, Clément L. Canonne, and Joy Qiping Yang. Independence testing for bounded degree Bayesian networks. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022 (NeurIPS 2022)*, 2022.

Wray L. Buntine. Theory refinement on Bayesian networks. In *UAI '91: Proceedings of the Seventh Annual Conference on Uncertainty in Artificial Intelligence, University of California at Los Angeles, Los Angeles, CA, USA, July 13-15, 1991*, pp. 52–60. Morgan Kaufmann, 1991.

Jie Cheng and Russell Greiner. Comparing Bayesian network classifiers. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI 1999)*, pp. 101–108, 1999.

David Maxwell Chickering. Learning Bayesian networks is NP-complete. In *Learning from Data: Artificial Intelligence and Statistics V (AISTATS 1995)*, pp. 121–130. Springer, 1995.

Davin Choo, Joy Qiping Yang, Arnab Bhattacharyya, and Clément L. Canonne. Learning bounded-degree polytrees with known skeleton. In *International Conference on Algorithmic Learning Theory (ALT 2024)*, Proceedings of Machine Learning Research, pp. 402–443. PMLR, 2024.

Gregory F. Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Mach. Learn.*, 9:309–347, 1992.

James Cussens. GOBNILP: learning Bayesian network structure with integer programming. In *International Conference on Probabilistic Graphical Models, (PGM 2020)*, Proceedings of Machine Learning Research, pp. 605–608. PMLR, 2020.

Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

Abhimanyu Das and David Kempe. Approximate submodularity and its applications: Subset selection, sparse approximation and dictionary selection. *J. Mach. Learn. Res.*, 19:3:1–3:34, 2018.

Sanjoy Dasgupta. Learning polytrees. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI 1999)*, pp. 134–141. Morgan Kaufmann, 1999.

Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theor. Comput. Sci.*, 141(1&2):109–131, 1995a.

Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM J. Comput.*, 24(4):873–921, 1995b.

Jack Edmonds. Optimum branchings. *J. Res. Natl. Bur. Stand. B*, 71(4):233–240, 1967.

Michael Etscheid and Heiko Röglin. Smoothed analysis of local search for the maximum-cut problem. *ACM Trans. Algorithms*, 13(2):25:1–25:12, 2017.

Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI 1993)*, pp. 1022–1029, 1993.

Hamid Fehri, Ali Gooya, Yuanjun Lu, Erik Meijering, Simon A. Johnston, and Alejandro F. Frangi. Bayesian polytrees with learned deep features for multi-class cell segmentation. *IEEE Trans. Image Process.*, 28(7):3246–3260, 2019.

Eibe Frank, Mark A. Hall, and Ian H. Witten. The WEKA workbench. `https://ml.cms.waikato.ac.nz/weka/Witten_et_al_2016_appendix.pdf`, 2016. Online Appendix for *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, Fourth Edition, 2016.

Robert Ganian and Viktoriia Korchemna. The complexity of Bayesian Network Learning: Revisiting the superstructure. In *Proceedings of the 35th Annual Conference on Neural Information Processing Systems (NeurIPS 2021)*, pp. 430–442, 2021.

Robert Ganian and Viktoriia Korchemna. The complexity of Bayesian Network Learning: Revisiting the superstructure. *CoRR*, abs/2602.10253, 2026.

M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

Serge Gaspers, Mikko Koivisto, Mathieu Liedloff, Sebastian Ordyniak, and Stefan Szeider. On finding optimal polytrees. *Theor. Comput. Sci.*, 592:49–58, 2015.

Curtis Greene and Thomas L Magnanti. Some abstract pivot algorithms. *SIAM Journal on Applied Mathematics*, 29(3):530–539, 1975.

Niels Grüttemeier and Christian Komusiewicz. Learning Bayesian networks under sparsity constraints: A parameterized complexity analysis. *J. Artif. Intell. Res.*, 74:1225–1267, 2022.

Niels Grüttemeier, Christian Komusiewicz, and Nils Morawietz. On the parameterized complexity of polytree learning. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI 2021)*, pp. 4221–4227. ijcai.org, 2021a.

Niels Grüttemeier, Christian Komusiewicz, and Nils Morawietz. Efficient Bayesian network structure learning via parameterized local search on topological orderings. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2021)*, pp. 12328–12335. AAAI Press, 2021b.

John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society*, 28(1):100–108, 1979.

Juha Harviainen, Frank Sommer, and Manuel Sorge. Exact and approximate algorithms for polytree learning. *CoRR*, abs/2605.03622, 2026.

David Heckerman, Dan Geiger, and David Maxwell Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Mach. Learn.*, 20(3):197–243, 1995.

Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62 (2):367–375, 2001.

Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *J. Comput. Syst. Sci.*, 37(1):79–100, 1988.

Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The UCI machine learning repository. `https://archive.ics.uci.edu`, 2007. Accessed: 2026-01-27.

Neville Kenneth Kitson, Anthony C. Constantinou, Zhigao Guo, Yang Liu, and Kiattikun Chobtham. A survey of Bayesian network structure learning. *Artif. Intell. Rev.*, 56(8):8721–8814, 2023.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009.

Madhumita Kundu, Pekka Parviainen, and Saket Saurabh. Time–approximation trade-offs for learning Bayesian networks. In *Proceedings of The 12th International Conference on Probabilistic Graphical Models (PGM 2024)*, volume 246 of *Proceedings of Machine Learning Research*, pp. 486–497. PMLR, 2024a.

Madhumita Kundu, Pekka Parviainen, and Saket Saurabh. Discovering Bayesian networks when few variables matter. In *Proceedings of the 27th European Conference on Artificial Intelligence (ECAI 2024)*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, pp. 3047–3054. IOS Press, 2024b.

Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Math. Oper. Res.*, 35(4):795–806, 2010.

Daniele Nikzad, Alexander Zhilkin, Juha Harviainen, Jack Kuipers, Giusi Moffa, and Mikko Koivisto. Scaling up Bayesian DAG sampling. *CoRR*, abs/2510.25254, 2025. doi: 10.48550/ARXIV.2510.25254.

Sebastian Ordyniak and Stefan Szeider. Parameterized complexity results for exact Bayesian network structure learning. *J. Artif. Intell. Res.*, 46:263–302, 2013.

James G. Oxley. *Matroid theory*. Oxford University Press, 1992. ISBN 978-0-19-853563-8.

Judea Pearl. *Probabilistic reasoning in intelligent systems - networks of plausible inference*. Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann, 1989.

Mauro Scanagatta, Cassio P. de Campos, Giorgio Corani, and Marco Zaffalon. Learning Bayesian networks with thousands of variables. In *Advances in Neural Information Processing Systems 28 (NeurIPS 2015)*, pp. 1864–1872, 2015.

Mauro Scanagatta, Giorgio Corani, Cassio P. de Campos, and Marco Zaffalon. Learning treewidth-bounded Bayesian networks with thousands of variables. In *Advances in Neural Information Processing Systems 29 (NeurIPS 2016)*, pp. 1462–1470, 2016.

Alexander Schrijver et al. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

Maxim Sviridenko, Jan Vondrák, and Justin Ward. Optimal approximation for submodular and supermodular optimization with bounded curvature. *Math. Oper. Res.*, 42(4):1197–1218, 2017.

Andrea Vattani. k-means requires exponentially many iterations even in the plane. *Discret. Comput. Geom.*, 45(4):596–616, 2011.

Kai Wei, Rishabh K. Iyer, and Jeff A. Bilmes. Submodularity in data subset selection and active learning. In *Proceedings of the 32nd International Conference on Machine Learning, (ICML 2015)*, JMLR Workshop and Conference Proceedings, pp. 1954–1963. JMLR.org, 2015.

Changhe Yuan and Brandon M. Malone. Learning optimal Bayesian networks: A shortest path perspective. *J. Artif. Intell. Res.*, 48:23–65, 2013.

Valentin Ziegler. Approximation algorithms for restricted Bayesian network structures. *Inf. Process. Lett.*, 108(2):60–63, 2008.

A Additional Material for Section 2

If k is not a constant we use the following efficient encoding to avoid the 2^k overhead implied by the subsets of a maximal parent set: We only store the *minimal marginal gains* which roughly speaking provides an efficient encoding to compute the score of all parent sets in polynomial time. For a node v and a candidate parent u, let

$$M_u^\star := \{B \subseteq N \setminus \{u, v\} : \mathsf{mG}(u \mid B') \neq mg(u \mid B) \text{ for every } B' \subseteq B\}$$

and let M_u store $\mathsf{mG}(u \mid B)$ for each $B \in M_u^\star$. Now the marginal gain of u with respect to *any* set A is computed as follows: $\mathsf{mG}(u \mid A) \coloneqq \mathsf{mG}(u \mid B_A)$ where $B_A \in M_u^\star$ such that $B_A \subseteq A$ such that $\mathsf{mG}(u \mid B_A)$ is minimal, that is, there is no $B_A^\star \subset B_A \subseteq A$ with $\mathsf{mG}(u \mid B_A^\star) < \mathsf{mG}(u \mid B_A)$. For submodular functions s_v this definition is sound. Note that monotonicity is not required.

Lemma A.1. *Let s_v be a submodular function. Then, for every $A \subseteq N \setminus \{u, v\}$ we have*

$$\min\{\mathsf{mG}(u \mid B) : B \in M_u^\star, B \subseteq A\} = \mathsf{mG}(u \mid A).$$

Consequently, for every set P and every ordering $u_1, \ldots, u_r$ of P we have $\sum_{i \in [r]} \mathsf{mG}(u_i \mid \{u_1, \ldots, u_{i-1}\}) = s_v(P)$.

Proof. For the $' \geq'$ direction, observe that submodularity implies that $\mathsf{mG}(u \mid \cdot)$ is non-increasing. Thus, every admissible set $B \subseteq A$ has $\mathsf{mG}(u \mid B) \geq \mathsf{mG}(u \mid A)$. For the $' \leq'$ direction, note that among the subsets of A we choose B to be inclusion-minimal with $\mathsf{mG}(u \mid B) = \mathsf{mG}(u \mid A)$. For every $B' \subseteq B$ we have $\mathsf{mG}(u \mid B') \geq \mathsf{mG}(u \mid B)$ due to submodularity and $\mathsf{mG}(u \mid B') \neq \mathsf{mG}(u \mid B)$ by the minimality of B. Hence, $\mathsf{mG}(u \mid B') > \mathsf{mG}(u \mid B)$. Thus, $B \in M_u^\star$ and it is admissible. The chain sum now telescopes to $s_v(P)$ independent of the order. □

Moreover, note that if the score functions are additive then we have $M_u^\star \coloneqq \{\emptyset\}$ since the existing arcs of a parent set are irrelevant. In other words, the minimal marginal gains are a way to measure closeness of a scoring function to being additive. With the help of the maximal parent sets and the minimal marginal gains the score of any parent set can be computed. We would like to note that we only rely on the minimal marginal gain representation for one of our hardness results (Theorem 3.3) since there the largest parent set k is unbounded. In this hardness result the number of maximal parent sets as well as the number of minimal marginal gains is polynomial in the number of nodes which then yields that the overall input size is polynomial in the number of nodes. In general, the number of minimal marginal gains might not be polynomial in the number of nodes.

B Additional Material for Section 3

B.1 An Example why Local Search is Necessary

Figure 1 shows that the simple greedy algorithm which always adds the highest-scoring parent set without violating acyclicity can be stuck in a bad local optimum.

B.2 Formal Definition of an Exchange Mapping

Formally, an *exchange mapping* f maps each parent set D_S^v of the optimal polytree S to an arc set $f(D_S^v)$ of the (approximate) polytree A of size at most $|D_S^v|$ such that

a) Disjointness is preserved, that is, for every two parent sets D_S^v and D_S^w of the optimal polytree with $v \neq w$ we have $f(D_S^v) \cap f(D_S^w) = \emptyset$, and

b) adding D_S^v to the (approximate) polytree A and removing the current parent set D_A^v of v in the (approximate) polytree A as well as the arcs of $f(D_S^v)$ yields a polytree, that is, $A_v^\star := \big(A \setminus (D_A^v \cup f(D_S^v))\big) \cup D_S^v$ is a polytree.

   Note that the polytree $A_v^\star$ and the polytree A_v' defined in Line 12 of Algorithm 1 differ in the sets $\mathsf{fes}(D_S^v)$ and $f(D_S^v)$, respectively.

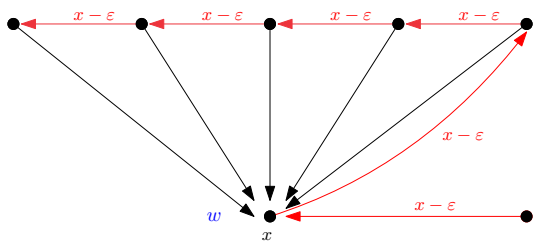


Figure 1: An example of a bad local optimum for the greedy algorithm which in each step adds the parent set with largest score which preserves acyclicity where $k = 5$. The score functions are as follows. The black parent set yields score x (more precisely, each black arc yields score x/k individually) while each red parent set yields score $x - \varepsilon$. Moreover, for node w, the score of ℓ black edges together with the incoming red edge is $(x - \varepsilon) + \varepsilon \cdot \ell/k$. Note that so-defined score function of each node is submodular.
The greedy algorithm can only add the black parent set and thus the resulting polytree has score x while the polytree consisting of all red arcs has score $6(x - \varepsilon)$. Extending this example to arbitrary k shows that the greedy algorithm only yields a $(k + 1)$ approximation.

B.3 General Steps of the 2-Approximation

Since A is locally optimal, it is not possible to add any parent set D_S^v of the optimal solution S in Algorithm 1. Hence, for the polytree

$$A_v := \big(A \setminus (D_A^v \cup \mathsf{fes}(D_S^v))\big) \cup D_S^v \tag{1}$$

we have $s(A) \geq s(A_v)$. In words, the two polytrees A and A_v differ as follows: (a) node v has parent set D_A^v in A and parent set D_S^v in A_v, and (b) in terms of edges with a different endpoint than v, the polytree A has additionally the arcs $\mathsf{fes}(D_S^v)$ compared to A_v.

Let f be an exchange mapping between an optimal polytree S and the approximate polytree A computed by our algorithm; later we argue that f exists. By definition, f has the following two properties: a) f preserves disjointness, that is, for any two parent sets D_S^v and D_S^w of the optimal polytree with $v \neq w$ we have $f(D_S^v) \cap f(D_S^w) = \emptyset$, and b) $A_v^\star := \big(A \setminus (D_A^v \cup f(D_S^v))\big) \cup D_S^v$ is a polytree. Observe that the polytrees A_v and $A_v^\star$ only differ in the arc sets $\mathsf{fes}(D_S^v)$ (missing in A_v) and $f(D_S^v)$ (missing in $A_v^\star$). Note that $\mathsf{fes}(D_S^v) \cap f(D_S^v) \neq \emptyset$ is possible. Also note that $f(D_S^v) \cap D_A^v \neq \emptyset$ is possible.

By definition of $\mathsf{fes}(D_S^v)$, for any arc set Z such that $(A \setminus (D_A^v \cup Z)) \cup D_S^v$ is a polytree, $\mathsf{fes}(D_S^v)$ is the cheapest one in terms of score. Moreover, note that since `LS-PT` only adds reasonable parent sets, we always have $|\mathsf{fes}(Z)| \leq |Z|$ as removing any subset of edges of any parent set can never increase the score. By property b) of the exchange mapping f, $f(D_S^v)$ is also a valid candidate for Z. Thus, we have

$$\begin{aligned} s(A_v) = \; & s\big(\big(A \setminus (D_A^v \cup \mathsf{fes}(D_S^v))\big) \cup D_S^v\big) && (2)\\ \geq \; & s\big(\big(A \setminus (D_A^v \cup f(D_S^v))\big) \cup D_S^v\big) \quad = s(A_v^\star). && (3) \end{aligned}$$

Consequently,

$$s(A) \geq s(A_v) \geq s(A_v^\star). \tag{4}$$

We now consider the arcs which are not in both polytrees A and $A_v^\star$: First observe that node v has parent set D_A^v in A_v and v has parent set D_S^v in $A_v^\star$. Note that $D_S^v \cap D_A^v \neq \emptyset$ is possible. Moreover, for any other node $w \neq v$ by Q_w^v we denote the set of incoming arcs of $f(D_S^v)$ to w. Note that $Q_w^v = \emptyset$ is possible, that is, whenever $f(D_S^v)$ does not contain an arc towards w. Let D_A^w be the parent set of w in A. By definition of Q_w^v node w has parent set $D_A^w \setminus Q_w^v$ in $A_v^\star$. By the definition of the marginal gain, we have $s(D_A^w) = s(D_A^w \setminus Q_w^v) + \mathsf{mG}(Q_w^v \mid (D_A^w \setminus Q_w^v))$. In other words, node $w \neq v$ contributes $\mathsf{mG}(Q_w^v \mid (D_A^w \setminus Q_w^v))$ more score to $s(A)$ than to $s(A_v^\star)$. Recall that the score of any polytree is the sum of the scores of its parent sets. Consequently, $s(A_v^\star) \leq s(A)$ implies

the following:

$$s(A_v^\star) \leq s(A) \tag{5}$$

$$\Leftrightarrow s(D_S^v) + \sum_{w \in N \setminus \{v\}} s(D_A^w \setminus Q_w^v) \leq s(D_A^v) + \sum_{w \in N \setminus \{v\}} s(D_A^w) \tag{6}$$

$$\Leftrightarrow s(D_S^v) \leq s(D_A^v) + \sum_{w \in N \setminus \{v\}} \mathsf{mG}(Q_w^v \mid (D_A^w \setminus Q_w^v)). \tag{7}$$

Now, we sum over all nodes of N, that is, we simultaneously consider the costs of replacing any parent set D_A^v in A by its corresponding parent set D_S^v in S. We obtain

$$\sum_{v \in N} s(D_S^v) \leq \sum_{v \in N} (s(D_A^v)) + \sum_{v \in N} \sum_{w \in N \setminus \{v\}} \mathsf{mG}(Q_w^v \mid (D_A^w \setminus Q_w^v)). \tag{8}$$

Since $\sum_{v \in N} s(D_S^v)$ and $\sum_{v \in N} s(D_A^v)$ are precisely the decomposition of the optimal solution S and the approximate solution A into their parent sets, respectively, we obtain

$$s(S) \leq s(A) + \sum_{v \in N} \sum_{w \in N \setminus \{v\}} \mathsf{mG}(Q_w^v \mid (D_A^w \setminus Q_w^v)). \tag{9}$$

Thus, it remains to upper bound

$$\sum_{v \in N} \sum_{w \in N \setminus \{v\}} \mathsf{mG}(Q_w^v \mid (D_A^w \setminus Q_w^v)) = \sum_{w \in N} \sum_{v \in N \setminus \{w\}} \mathsf{mG}(Q_w^v \mid (D_A^w \setminus Q_w^v))\,;$$

here the roles of v and w are swapped. Now, for any fixed w, fix an arbitrary ordering $v_1, \ldots, v_{n-1}$ of $N \setminus \{w\}$. Due to submodularity, we have

$$\mathsf{mG}(Q_w^{v_i} \mid D_A^w \setminus Q_w^{v_i}) \leq \mathsf{mG}(Q_w^{v_i} \mid \bigcup_{j=1}^{i-1} Q_w^{v_j}), \tag{10}$$

and by the definition of marginal gains, we have

$$s(\bigcup_{j=1}^{i-1} Q_w^{v_j}) + \mathsf{mG}(Q_w^{v_i} \mid \bigcup_{j=1}^{i-1} Q_w^{v_j}) = s(\bigcup_{j=1}^{i} Q_w^{v_j}), \tag{11}$$

and thus we obtain

$$\sum_{v \in N \setminus \{w\}} \mathsf{mG}(Q_w^v \mid (D_A^w \setminus Q_w^v)) \leq \sum_{i=1}^{n-1} \mathsf{mG}(Q_w^{v_i} \mid \bigcup_{j=1}^{i-1} Q_w^{v_j}) \tag{12}$$

$$= s(\bigcup_{j=1}^{n-1} Q_w^{v_j}) - s_w(\emptyset) = s(\bigcup_{j=1}^{n-1} Q_w^{v_j}) \tag{13}$$

$$\leq s(D_A^w). \tag{14}$$

Here, the first inequality follows from Equation (10) and the first equality follows from Equation (11). Moreover, note that by definition $s_w(\emptyset) = 0$. Also, Equation (14) follows from f being monotone. Finally observe that Equation (14) can be strict since the $Q_w^{v_j}$ do not necessarily form a partition of D_A^w.

Consequently, we obtain

$$\sum_{w \in N} \sum_{v \in N \setminus \{w\}} \mathsf{mG}(Q_w^v \mid (D_A^w \setminus Q_w^v)) \leq \sum_{w \in N} s(D_A^w) = s(A). \tag{15}$$

Hence, we have proven that A is indeed a 2-approximation of S.

B.4 CONSTRUCTION OF AN EXCHANGE MAPPING f

We now construct the desired exchange mapping f between the optimal polytree S and the approximate polytree A; for a visualization we refer to Figure 2. By $\overline{S}$ and $\overline{A}$ we denote the underlying trees of S and A, respectively. First, we add *dummy edges* to $\overline{S}$ and $\overline{A}$ (by abusing notation we call the resulting trees also $\overline{S}$ and $\overline{A}$, respectively) such that both form a spanning tree on the node set N. By $\widetilde{D'_S}$ and $\widetilde{D'_A}$ we denote the sets of added dummy edges, respectively. We do this greedily by adding arbitrary edges.

We now consider the so-called *graphical matroid* $\mathcal{G}$ of forest on N. We now present the necessary definitions; for more details on matroids we refer to the standard monographs (Oxley, 1992; Schrijver et al., 2003).

Definition B.1. A *matroid* is a pair $(\mathcal{E}, \mathcal{I})$ consisting of a finite set $\mathcal{E}$ and a set of subsets $\mathcal{I}$ of $\mathcal{E}$, denoted as *independent sets*, such that

1. $\emptyset \in \mathcal{I}$,
2. $J \in \mathcal{I}$ and $I \subseteq J$, then also $I \in \mathcal{I}$, and
3. for $I_1, I_2 \in \mathcal{I}$ with $|I_1| < |I_2|$ there exists an element $e \in I_2 \setminus I_1$ such that $I_1 \cup \{e\} \in \mathcal{I}$.

The maximal independent sets of $\mathcal{I}$ are called *bases*. All bases of a matroid have the same size.

One example of a matroid is the graphical matroid which consists of all forests on a node set.

Fact B.2 ((Oxley, 1992; Schrijver et al., 2003))**.** For a node set N, let $E(N)$ be the set of all edges on N and let $\mathcal{T}$ be the set of all forests on N. Then $(E(N), \mathcal{T})$ is a matroid, the so-called *graphical matroid* and its bases are the spanning trees on N.

We make use of a so-called *generalized Rota-exchange property* (Greene & Magnanti, 1975; Schrijver et al., 2003; Lee et al., 2010). Intuitively, we are given two bases P and Q of a matroid and a family $P_1, P_2, \dots, P_\ell$ of disjoint subsets of P, then this property yields that also a family $Q_1, Q_2, \dots, Q_\ell$ of disjoint subsets of Q exists such that in base Q the set Q_i can be replaced by the set P_i and we still have a base.

Lemma B.3 ((Lee et al., 2010, Lemma 2.6)(Greene & Magnanti, 1975, Theorem 3.3) (Schrijver et al., 2003, Equation 42.15))**.** *Let $(\mathcal{E}, \mathcal{I})$ be a matroid and let P, Q be bases of $(\mathcal{E}, \mathcal{I})$. Moreover, let $P_1, P_2, \dots, P_\ell$ be subsets of P such that each element of P appears in exactly one P_i. Then there exist subsets $Q_1, Q_2, \dots, Q_\ell$ of Q such that each element of Q appears in exactly one of them, and for each $i \in [\ell]$ we have that $P_i \cup (Q \setminus Q_i)$ is a basis of $(\mathcal{E}, \mathcal{I})$.* [2]

In the terminology of a graphical matroid Lemma B.3 means that given two spanning trees $\overline{S}$ and $\overline{A}$ on a node set N, for any partition of the edges of $\overline{S}$ into $\overline{S_1}, \dots, \overline{S_\ell}$, we can partition the edges of $\overline{A}$ into $\overline{A_1}, \dots, \overline{A_\ell}$ such that $\overline{S_i} \cup (\overline{A} \setminus \overline{A_i})$ is a spanning tree on N for each $i \in [\ell]$.

Since this statement holds for *any* partition of $\overline{S}$, we can choose one specific partition which we call the *parent set partition* where we basically partition the edges of $\overline{S}$ into the parent sets of the optimal solution S: For each parent set D_S^v of the optimal polytree S, by $\widetilde{D_S^v}$ we denote the corresponding set of edges in $\overline{S}$. Recall that by $\widetilde{D'_S}$ we denote the set of all dummy arcs of $\overline{S}$. Then, the generalized Rota-exchange property yields a mapping $\widetilde{f}$ which maps each $\widetilde{D_S^v}$ and $\widetilde{D'_S}$ to an edge set $f(\widetilde{D_S^v})$ and $f(\widetilde{D'_S})$ of $\overline{A}$, respectively, such that (a) every edge of $\overline{A}$ appears in exactly one set $f(\widetilde{D_S^v})$ or $f(\widetilde{D'_S})$, and (b) $\widetilde{D_S^v} \cup (A' \setminus f(\widetilde{D_S^v}))$ and $\widetilde{D'_S} \cup (A' \setminus \widetilde{f}(\widetilde{D'_S}))$ are all spanning trees of N.

Now, based on $\widetilde{f}$, we construct the exchange mapping f as follows. The basic idea is to ignore the dummy arcs of $(\widetilde{D'_A})$ in the image of any parent set of the optimal polytree. Formally, for each parent set D_S^v we set $f(D_S^v) := h\big(\widetilde{f}(\widetilde{D_S^v}) \setminus D'_A\big)$, where $h(\cdot)$ maps an edge of $\overline{A}$ to its corresponding arc

[2] Greene & Magnanti (1975) showed $Q_i \cup (P \setminus P_i) \in \mathcal{I}$. In Footnote 3 of Greene & Magnanti (1975) it is mentioned that $Q_i \cup (P \setminus P_i) \in \mathcal{I}$ can be shown via an inductive argument using the Rota-exchange property (given bases P, Q and $P' \subseteq P$ there exists $Q' \subseteq Q$ such that *both* $(P \setminus P') \cup Q'$ and $(Q \setminus Q') \cup P'$ are bases). This inductive proof also works analogously for showing that $P_i \cup (Q \setminus Q_i) \in \mathcal{I}$.

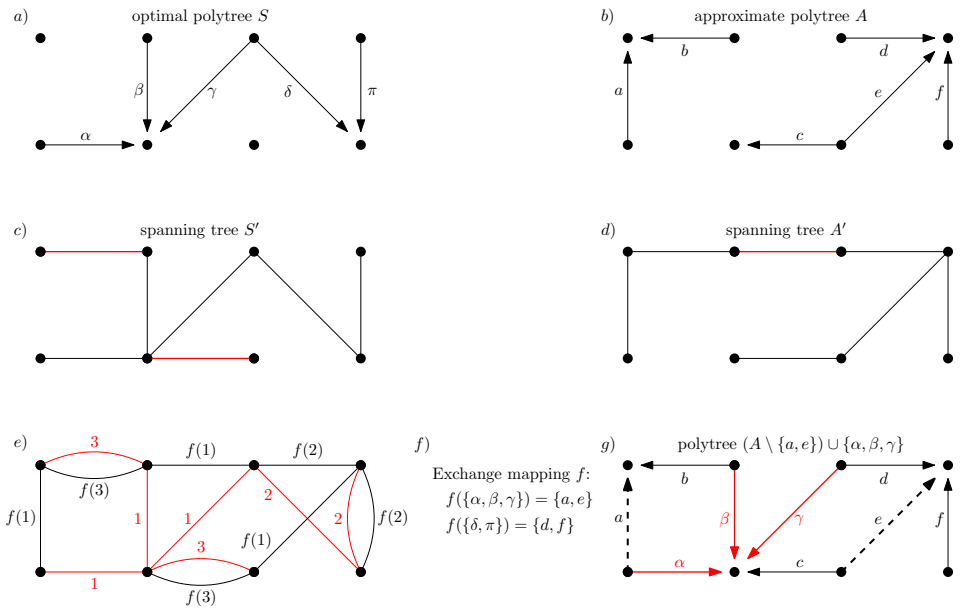


Figure 2: Visualization of the existence of an exchange mapping f. Parts $a)$ and $b)$ show the optimal polytree S and the approximate polytree A. Here, the property of A being locally optimal is dropped for ease of visualization. Parts $c)$ and $d)$ show the resulting spanning trees $\overline{S}$ and $\overline{A}$ based on S and A, respectively. Here, the red correspond to the added dummy edges. Part $e)$ shows the result of the generalized Rota-exchange property (Lemma B.3). Here, the red numbers indicate the partition of $E(\overline{S})$ into the parent sets of S and into the dummy edges, and the black numbers indicate the partition of $E(\overline{A})$ obtained from the generalized Rota-exchange property. The resulting exchange mapping f is shown in $f)$. For the parent set $\{\alpha,\beta,\gamma\}$ polytree $(A \setminus \{a,e\}) \cup \{\alpha,\beta,\gamma\}$ is shown in $g)$. Here, the added parent set $\{\alpha,\beta,\gamma\}$ is shown in red and the removed arcs $\{a,e\}$ of A are drawn dashed.

of the approximate polytree A. Note that $f(D_S^v)$ is well-defined since all dummy arcs $\widetilde{D'_A}$ of $\overline{A}$ got removed. It remains to argue that f is indeed an exchange mapping, that is, we need to show that properties a) and b) hold.

To see that a) holds, that is, that f preserves disjointness, note that $\widetilde{f}$ preserves disjointness and to construct f only dummy arcs of $\widetilde{f}$ got removed. Consequently, also f preserves disjointness.

It remains to show that b) holds, that is, that $A_v^\star := \big(A \setminus (D_A^v \cup f(D_S^v))\big) \cup D_S^v$ is a polytree for each parent set D_S^v of the optimal polytree S. By the generalized Rota-exchange property, we know that $\widetilde{A'} := (\overline{A} \setminus \widetilde{f}(\widetilde{D_S^v})) \cup \widetilde{D_S^v}$ is a spanning tree on N. Observe that removing all dummy edges $\widetilde{D'_A}$ of $\overline{A}$ from $\widetilde{A'}$ yields a forest $\widehat{A'}$ on N, that is

$$\big(\overline{A} \setminus \big(\widetilde{D'_A} \cup \widetilde{f}(\widetilde{D_S^v})\big)\big) \cup \widetilde{D_S^v} \tag{16}$$

$$= \big((\overline{A} \setminus \widetilde{D'_A}) \setminus \widetilde{f}(\widetilde{D_S^v})\big) \cup \widetilde{D_S^v} = \widehat{A'} \tag{17}$$

is a forest on N. By definition, forest $\widehat{A'}$ contains no dummy arc of $\widetilde{D'_A}$ and thus $\widehat{A} := h(\widehat{A'})$ is well-defined and is a sub-polytree of A (recall that $h(\cdot)$ maps each edge of $\overline{A}$ to its corresponding arc of the approximate polytree A). More precisely, $A = h(A' \setminus \widetilde{D'_A})$, $f(D_S^v) = h\big(\widetilde{f}(\widetilde{D_S^v} \setminus \widetilde{D'_A})\big)$, and $D_S^v = h(\widetilde{D_S^v})$. In other words,

$$\widehat{A} = \big(A' \setminus f(D_S^v)\big) \cup D_S^v \tag{18}$$

is a polytree on a node set N. Since $A_v^\star = \big(A \setminus (D_A^v \cup f(D_S^v))\big) \cup D_S^v$ is a subset of $\widehat{A}$, we hence conclude that $A_v^\star$ is a polytree on N as well and thus property b) is verified. Hence, Theorem 3.1 is proven.

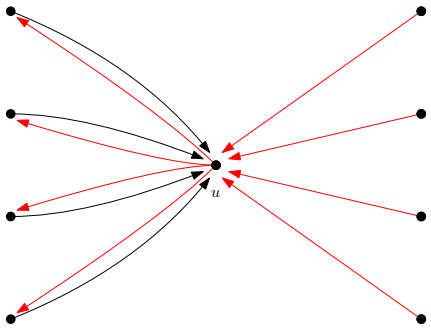


Figure 3: An example for a bad local optima where $k = 4$. Every arc yields a score of 1. Thus, the scoring function of u is additive. Hence, both the black and red parent set of u yield a score of 4 each. The polytree T consisting of the black arcs with score 4 is a local optima found by our algorithm LS−PT. The polytree $T^\star$ consisting of the red arcs has a score of 8 and thus LS−PT cannot achieve an approximation factor better than 2.

Assumption of f being monotone. The fact that f is monotone is only used in Equation (14). It is, however, possible to argue the correctness of Equation (14) by using only the local optimality of A. In other words, Theorem 3.1 also holds for variants of $\text{PT}^{\text{sm}}_{\leq}$ and PT^{sm} where we only impose the requirement that f is submodular (and not also monotone). For this, we need the following statement about subset-optimality of a local optimum.

Lemma B.4. *Let A be a polytree computed by LS−PT (Algorithm 1). Then, for every node $w \in N$ and every $U \subseteq D^w_A$, we have $s_w(U) \leq s_w(D^w_A)$.*

Proof. By definition of LS−PT (Algorithm 1), there exists a maximal parent set $D^\star_w \in \mathcal{F}_w$ such that $D^w_A \subseteq D^\star_w$. Thus, $U \subseteq D^\star_w$ and U is one of the candidates which are enumerated in Line 10 for the node $v := w$. Set

$$A' := (A \setminus D^w_A) \cup U.$$

Since $U \subseteq D^w_A$, we have $A' \subseteq A$ which is clearly a polytree which also respects a possible maximum in-degree bound of k. Thus, $Z := \emptyset$ is an admissible arc removal set in Line 11 and the polytree A_w that Line 11 constructs for the candidate set U satisfies $s(A_w) \geq s(A')$. But since LS−PT returned polytree A, we know that the strict test in Line 12 failed for every candidate, that is, in particular $s(A) \geq s(A_w)$. Finally, polytrees A and A' only differ in the parent set of node w, so by the score decomposition we have $s(A') = s(A) - s_w(D^w_A) + s_w(U)$. This yields

$$s(A) \geq s(A_w) \geq s(A') = s(A) - s_w(D^w_A) + s_w(U),$$

which yields the statement. □

B.5 Limits of LS−PT (Algorithm 1)

For an example that the approximation guarantee of 2 for LS−PT (Algorithm 1) on monotone and submodular scoring functions is tight we refer to Figure 3.

Recall that a variant of Algorithm 1 yields a $(k + 1)$ approximation for the general polytree learning problem $\text{PT}_{\leq}$ without any further constraints (Harviainen et al., 2026, Theorem 4.1), that is, the greedy algorithm adds the highest scoring parent set to the polytree which preserves acyclicity. One could hope that our local search algorithm Algorithm 1 can be used to obtain a better approximation guarantee than $k + 1$ for $\text{PT}_{\leq}$. An adaption of the example in Figure 1, however, shows that this is not possible: In contrast to the example in Figure 1 assume that each parent set of w containing at least one, but not all black edges yields score 0. Then after the greedy algorithm added the black parent set it has reached a local optimum since to add any red parent set the entire black parent set needs to be removed and thus the score would decrease. Note that this issue of being stuck in such a bad local optima does not occur if we assume the score functions to be submodular since then every subset of the black parent set needs to have a non-zero score and it becomes possible to remove only parts of the black parent set without losing its entire score.

B.6 Proof of Theorem 3.2

Theorem 3.2 (★). *Each iteration of* `LS-PT` *(Algorithm 1) can be done in* $4^k \cdot |\mathcal{F}| \cdot \mathcal{O}(n)$ *time, where* $\mathcal{F}$ *denotes the set of maximal parent sets and* k *the largest maximal parent set size.*

We first need some intermediate results.

Lemma B.5. *Let* A *be a polytree on* N *with parent sets* $(D_A^y)_{y\in N}$ *and let* $k := \max_{y\in N} |D^y|$*. After a preprocessing of* A *taking* $\mathcal{O}(2^k \cdot k \cdot n)$ *time, the following can be done in* $\mathcal{O}(2^k \cdot n)$ *time for each pair consisting of a node* v *and a parent set* $P_v \subseteq N \setminus \{v\}$ *pair: Let* $A^\star := A \setminus D_A^v$ *and for an arc set* X *let* $g(X) := s(A^\star) - s(A^\star \setminus X)$*. Then, one can compute a set*

$$\mathsf{fes}_A^v(P_v) \in \arg\min\{g(X) : X \subseteq A^\star, (A^\star \setminus X) \cup P_v \text{ is a polytree}\}$$

together with its value $g(\mathsf{fes}_A^v(P_v))$*.*

Proof. By $F^\star$ we denote the skeleton of $A^\star$, and by $T := \{v\} \cup P_v$ we denote the set of *terminals*. We write $D_\star^y$ for the parent set of y in $A^\star$. Thus, $D_\star^y = D_A^y$ for $y \neq v$ and $D_\star^v = \emptyset$. Moreover, a *removal set* $X \subseteq A^\star$ is a set of arcs, and we split it according to the *head* of each arc: for $y \in N$,

$$X_y \;:=\; \{\, (p, y) \in X \,\} \qquad \text{which are the arcs of } X \text{ that point } \textit{into } y.$$

Every arc of $A^\star$ has exactly one head, so the sets $(X_y)_{y\in N}$ partition X. Since the arcs pointing into y are exactly $\{(p, y) : p \in D_\star^y\}$, we have $X_y \subseteq D_\star^y$. Note that we always have $X_v = \emptyset$ as $A^\star$ was formed by deleting D_A^v from A. Moreover, for $y \neq v$ we have $X_y = \emptyset$ whenever y has no parents in $A^\star$.

Step 1: The problem is a multiway cut in a forest.

Claim B.6. For $X \subseteq A^\star$, the digraph $(A^\star \setminus X) \cup P_v$ is a polytree if and only if every component of $F^\star - X$ contains at most one terminal.

Proof of Claim. Since $A^\star$ contains no arc with head v, we have $A^\star \cap D_A^v = \emptyset$, and the skeleton of $(A^\star \setminus X) \cup D_A^v$ arises from $F^\star - X$ by adding the edges $\{u, v\}$ for $u \in D_A^v$. Insert these edges one at a time. All of them are incident with v, so after inserting some of them the endpoint v lies in the union of the component of v and the components of the already inserted parents. Hence some insertion closes a cycle if and only if two terminals lie in the same component of $F^\star - X$. Finally, a digraph is a polytree if and only if its skeleton is a forest, so no additional condition on the orientations arises. ■

Consequently, $\mathsf{fes}_A^v(P_v)$ is a *minimum-cost multiway cut separating* T *in the forest* $F^\star$.

Step 2: The cost decomposes over heads.

Claim B.7. Let $c_y(Y) := s_y(D_\star^y) - s_y(D_\star^y \setminus Y)$ for $Y \subseteq D_\star^y$ which are the cost of removing the arcs Y from the parent set of y. Then, for every $X \subseteq A^\star$, we have $g(X) = \sum_{y\in N} c_y(X_y)$.

Proof of Claim. The parent set of y in $A^\star \setminus X$ is $D_\star^y \setminus X_y$, so $s(A^\star) - s(A^\star \setminus X) = \sum_y \big(s_y(D_\star^y) - s_y(D_\star^y \setminus X_y)\big)$. ■

Note that c_y is *super-additive* since applying submodularity of s_y to $D_\star^y \setminus \{a\}$ and $D_\star^y \setminus \{b\}$ gives $s_y(D_\star^y - a) + s_y(D_\star^y - b) \geq s_y(D_\star^y) + s_y(D_\star^y - a - b)$ and hence $c_y(\{a\}) + c_y(\{b\}) \leq c_y(\{a, b\})$.

Step 3: Rooting, and ownership of edges.

Rooting. Root every tree of $F^\star$ at an arbitrary node. For $y \in N$ let $p(y)$ denote the parent of y in that rooting (undefined if y is a root), let $\mathrm{ch}(y)$ denote the set of children of y, and let T_y denote the set of nodes of the subtree rooted at y, so that $F^\star[T_y]$ is that subtree.

Ownership. Each edge of $F^\star$ is the skeleton of exactly one arc of $A^\star$. We say that this arc is *owned* by that arc's head. Thus, y owns exactly the edges $\{y, p\}$ with $p \in D_\star^y$, and choosing a set $X_y \subseteq D_\star^y$ for every y decides the status (removed or kept) of every edge of $F^\star$ exactly once. Note that the choices are independent. In particular, the arcs with head in T_y are precisely the edges of $F^\star[T_y]$

together with $\{y, p(y)\}$ in the case that the latter is owned by y and the only edge of $F^\star$ leaving T_y is $\{y, p(y)\}$, which is what makes the components of $F^\star[T_y] - X$ other than the one containing y already final.

Step 4: The dynamic program.

The idea behind the table. We work through the forest bottom-up, settling at each node y which of the arcs entering y are cut. Once the subtree T_y has been processed, every component of $F^\star[T_y] - X$ except the one containing y is final. Recall that by Step 3 the only edge leaving T_y is $\{y, p(y)\}$. Thus, those components can be checked against the terminal condition immediately and can then be forgotten, and all that the rest of the forest still needs to know about T_y is carried by two bits: The first, ℓ, records whether the one component that is still open (the one containing y) already holds a terminal. Note that one bit for the number of components suffices, because that component can only grow as we move upward, so two terminals in that component are irrecoverably infeasible. The second bit β records whether the edge $\{y, p(y)\}$ has been cut. This bit exists only because that edge may be *owned* by y. In this case y decides it and pays for it, and the parent has to be told.

Formal definition of the table. For $\ell, \beta \in \{0, 1\}$ let $\mathcal{X}_y(\ell, \beta)$ be the family of all sets X of arcs of $A^\star$ with head in T_y such that

1. every component of $F^\star[T_y] - X$ not containing y has at most one terminal;
2. the component of y in $F^\star[T_y] - X$ contains exactly ℓ terminals;
3. if y owns $\{y, p(y)\}$ then we have the equivalence $\{y, p(y)\} \in X \iff \beta = 1$; otherwise, and for a root, β is unconstrained,

and put $\mathrm{dp}_y[\ell][\beta] := \min\{\sum_{z \in T_y} c_z(X_z) : X \in \mathcal{X}_y(\ell, \beta)\}$ with $\min \emptyset := \infty$. Here, we implicitly use Claim B.7. Observe that when y does not own the edge $\{y, p(y)\}$, condition 3. is vacuous, so $\mathcal{X}_y(\ell, 0) = \mathcal{X}_y(\ell, 1)$ and $\mathrm{dp}_y[\ell][0] = \mathrm{dp}_y[\ell][1]$. The recurrence uses this.

Recurrence. Fix y and a set $X_y \subseteq D^y_\star$. For a child c put

$$\mathrm{cut}_c := \min\{\mathrm{dp}_c[0][1], \mathrm{dp}_c[1][1]\}, \qquad \mathrm{keep}^0_c := \mathrm{dp}_c[0][0], \qquad \mathrm{keep}^1_c := \mathrm{dp}_c[1][0],$$

Intuitively, cut_c is the cheapest solution for the subtree T_c in which the edge $\{y, c\}$ is *removed*, so that c's component is closed off there and then and may hold either 0 or 1 terminal. Moreover, keep^0_c is the cheapest solution in which that edge is *kept* (c's component will merge into the one containing y) and carries *no* terminal into it. Finally, keep^1_c is the cheapest such solution carrying *one* terminal into it. Note that the index $\beta = 1$ in cut_c and $\beta = 0$ in $\mathrm{keep}^0_c, \mathrm{keep}^1_c$ is exactly the record of whether the edge was removed, and it is immaterial when c does not own the edge, by the second observation above. Now let a_c be the cheapest way for c's side to contribute *no* terminal to the component of y, and b_c the cheapest way to contribute *one*. We now have:

$$\begin{aligned}
&\{y, c\} \text{ owned by } y,\ c \in X_y: && a_c = \mathrm{cut}_c, && b_c = \infty,\\
&\{y, c\} \text{ owned by } y,\ c \notin X_y: && a_c = \mathrm{keep}^0_c, && b_c = \mathrm{keep}^1_c,\\
&\{y, c\} \text{ owned by } c: && a_c = \min\{\mathrm{cut}_c, \mathrm{keep}^0_c\}, && b_c = \mathrm{keep}^1_c.
\end{aligned}$$

Note that the three cases are exhaustive, and in the first two the index β of c is immaterial by the second observation above. With the convention $\infty + x = \infty$, put

$$S_0 := \sum_{c \in \mathrm{ch}(y)} a_c, \qquad S_1 := \begin{cases} S_0, & y \in T,\\ \min\limits_{c \in \mathrm{ch}(y)} \Big(b_c + \sum\limits_{c' \neq c} a_{c'}\Big), & y \notin T, \end{cases}$$

and set, for each β consistent with X_y in the sense of (3),

$$\mathrm{dp}_y[\ell][\beta] \;=\; \min_{X_y} \big(c_y(X_y) + S_\ell\big), \qquad \text{with } \mathrm{dp}_y[0][\beta] = \infty \text{ when } y \in T.$$

Base case: for a leaf, $\mathrm{ch}(y) = \emptyset$, so $S_0 = S_1 = 0$ if $y \in T$ and $S_0 = 0$, $S_1 = \infty$ otherwise; $D^y_\star \subseteq \{p(y)\}$, and the formula reduces to $\mathrm{dp}_y[y][\beta] = c_y(X_y)$ for the unique X_y compatible with β for each $y \in T$.

Correctness. We use induction on $|T_y|$. By Step 3 the arcs with head in T_y split disjointly into X_y and, for each child c, the arcs with head in T_c. Claim B.7 lets us deal with children independently, once X_y is fixed. Moreover, the components of $F^\star[T_y] - X$ are the components of $F^\star[T_c] - X$ not containing c, together with the component of y, which consists of y and the components of those children c whose edge $\{y, c\}$ was kept. Hence, condition 1. for y is condition 1. for the children plus condition 2. for the children whose edge was cut, and the terminal count of y's component is $[y \in T] + \sum_{c \text{ kept}} \ell_c$, which must not exceed 1. Here $[y \in T]$ is 1 if $y \in T$ and it is 0 otherwise, and ℓ_c is the first index of the entry of c's table that is used (the number of terminals in the still-open component of c inside T_c). That is exactly what a_c, b_c, S_0 and S_1 encode. Conversely every combination of child solutions recombines into a member of $\mathcal{X}_y(\ell, \beta)$, so the minimum is attained.

Output. Since a root owns no parent edge, condition 3. is vacuous there and

$$g\big(\mathsf{fes}_A^v(P_v)\big) \;=\; \sum_{r \text{ root}} \min\{\mathrm{dp}_r[0][0],\ \mathrm{dp}_r[1][0]\},$$

by Claim B.6 applied to each tree separately, and the set itself is recovered by standard backtracking.

Step 5: Running time.

The preprocessing tabulates $c_y(Y)$ for all $Y \subseteq D_A^y$ and all y: these are at most 2^κ values per node, each yielding one score evaluation. Hence, we need $\mathcal{O}(2^\kappa \kappa)$ time per node and $\mathcal{O}(2^\kappa \kappa n)$ time in total. These tables serve *every* candidate of the iteration: for $y \neq v$ the parent set in $A^\star$ equals the one in A, and for $y = v$ we have $X_v = \emptyset$, so the table of v is never consulted.

For a fixed candidate, building $F^\star$ and rooting it costs $\mathcal{O}(n)$, and the dynamic program visits each y once, trying $2^{|D_\star^y|}$ sets X_y and spending $\mathcal{O}(|\mathrm{ch}(y)|)$ on each. Moreover, the quantities a_c, b_c are $\mathcal{O}(1)$ look-ups, S_0 is one sum, and S_1 is a leave-one-out minimum, evaluated by prefix and suffix sums. This is equivalent to counting the children with $a_c = \infty$: two or more give $S_1 = \infty$, exactly one (say c_0) gives $S_1 = b_{c_0} + \sum_{c \neq c_0} a_c$, and none gives $S_1 = S_0 + \min_c(b_c - a_c)$. Summing, $\sum_y 2^{|D_\star^y|}\mathcal{O}(|\mathrm{ch}(y)| + 1) = \mathcal{O}(2^\kappa n)$, since $\sum_y(|\mathrm{ch}(y)| + 1) = \mathcal{O}(n)$. □

We are now ready to prove the main claim.

Proof. An iteration considers at most $2^k \cdot |\mathcal{F}|$ candidates (v, D_v), since there are $|\mathcal{F}_v|$ choices of a maximal parent set of v and 2^k subsets of it. Every parent set of the current polytree A is a subset of a maximal parent set and hence has size at most k, so Lemma B.5 applies and yields $\mathsf{fes}(D_v)$, and with it $s(A_v)$, in $\mathcal{O}(2^k \cdot n)$ time per candidate after an $\mathcal{O}(2^k \cdot k \cdot n)$-time preprocessing that is performed once per iteration. □

B.7 Proof of Theorem 3.3

Theorem 3.3 (★). *Even if T has exactly one node with indegree at least two,* Best Improved Polytree *is NP-hard and cannot be solved in $2^{o(n)}$ time unless the ETH fails.*

Proof. We provide a reduction from Subcubic Independent Set. There, the input consists of a graph G where each vertex has degree at most 3, and an integer κ. The goal is to find an *independent set* $S \subseteq V(G)$, that is, no two vertices of S are adjacent, of size at least κ. Subcubic Independent Set is NP-hard (Garey & Johnson, 1979). Moreover, Subcubic Independent Set cannot be solved in $2^{o(n)}$ time unless the ETH fails (Impagliazzo et al., 2001). It is safe to assume that G contains no isolated vertices.

Construction. A visualization is provided in Figure 4. Our node set consists of two copies V_1 and V_2 of $V(G)$ and two further nodes p and q. Hence, we have $2|V| + 2$ nodes in total. Next, we provide a description scoring functions s_v for all $v \in N$. We only provide a description of the largest maximal parent sets (and some of there subsets, where convenient).

To define them, we make use of the constants $\alpha := 3n + 5, \beta := \alpha - 1, \tau := \alpha n - m - \kappa + 1, W := \tau + 1$, and $\rho := \frac{\tau}{n+1}$. Moreover, let $c := \tau n + n + 1$.

- For each node $v_1 \in V_1$, we set $f_{v_1}(v_2) := \beta$.

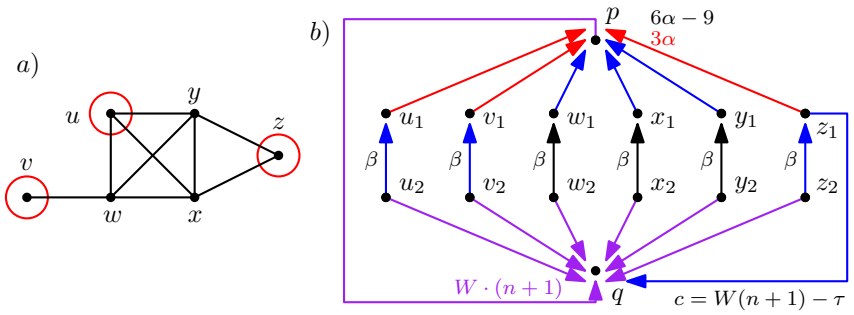


Figure 4: A visualization of the reduction from the proof of Theorem 3.3. For the input instance of SUBCUBIC INDEPENDENT SET with the graph shown in a) and $\kappa = 3$, the red vertices are a solution. Part b) shows the constructed BEST IMPROVED POLYTREE instance. The black numbers refer to the scores of the parent sets in the input polytree T, the violet arcs refer to the newly added parent set of node q in $T^\star$ which yields score $W(n+1)$, the blue arcs get removed from T to obtain $T^\star$, and the red arcs correspond the parent set of p in $T^\star$ which yields score 3α since $\{u, v, z\}$ are an independent set in G.

- Let $v_1 \in V_1$ be arbitrary but fixed. For any $Q \subseteq V_2 \cup \{p\}$, we set $f_q(Q) := W \cdot |Q|$, and we also set $f_q(Q \cup \{v_1\}) := c + \rho \cdot |Q|$.
- For any $W \subseteq V_1$ we set $f_p(W) := \alpha|W| - |E(G[W])|$.

Note that the unique maximal parent set of any node in V_2 is the empty parent set.

It is easy to see that f_{v_1} for each $v_1 \in V_1$ is submodular. It remains to argue that also f_q and f_p are submodular: Note that $\mathsf{mG}(x \mid Q) = W$ for each Q with $v_1 \notin Q$, and $\mathsf{mG}(x \mid Q) = \rho$ for each Q with $v_1 \in Q$. Consequently, no element has a larger marginal gain in a larger set. Furthermore, observe that $\mathsf{mG}(v_1 \mid Q) = c - (W - \rho)|Q \cap (V_2 \cup \{p\})|$ for each Q is non-increasing and non-negative. Thus, f_q is monotone and submodular. To see that f_p is monotone and submodular, observe that the function $|\cdot|$ is modular and that the function $|E(G[\cdot])|$ is supermodular. More precisely: $\mathsf{mG}(u \mid W) = 5 - |N(u) \cap W| \geq 2 > 0$ and these values never increase by larger W.

We now have to argue that the description of the scoring functions requires only polynomial size. Since every node in $V_1 \cup V_2$ only has a unique maximal parent set of size at most 1, the statement is clear for them and thus it remains to consider nodes p and q. For them, we bound both the largest maximal parent sets and the minimal marginal gains (see Section 2) by a polynomial in the input size. We first consider node q: The maximal parent sets are $\mathcal{F}_q = \{\{A\}, \{(A \setminus \{x\}) \cup \{v_1\} : x \in A\}\}$. For f_q we need the minimal marginal gains $\mathsf{mG}(v_1 \mid \emptyset)$, $\mathsf{mG}(x \mid \emptyset) = W$, and $\mathsf{mG}(x \mid \{v_1\}) = \rho$ for each $x \in V_2 \cup \{p\}$. These are $2n + 3$ many minimal marginal gains. Next, we consider node p: The unique maximal parent set is V_1. Observe that the minimal marginal gains of p are $\{\mathsf{mG}(u \mid B) = \alpha - |B| : B \subseteq N(u)\}$ for each $u \in V_1$. These are at most $8n$ minimal marginal gains. Consequently, the overall number of maximal parents sets as well as the overall number of minimal marginal gains is polynomial in the size of the number of nodes which in turn is polynomial in the size of G.

Next, we describe the initial polytree T. Node p has parent set V_1, each node $v_1 \in V_1$ has parent set $\{v_2\}$, and q has parent set $\{v_1\}$. Clearly, T is a polytree and only node p has indegree at least 2. Moreover, since G contains no isolated vertices, we have $s(T) := \beta \cdot n + \alpha \cdot n - m + c = 3n^3 + 11n^2 + 11n + 1 - m(n+1) - \kappa n$. Now, the aim is to find another polytree $T^\star$ which can be obtained from T by one iteration of `LS-PT`such that $s(T^\star) > s(T)$, that is, $s(T^\star) \geq \beta \cdot n + \alpha \cdot n - m + c + 1$. Clearly, the reduction can be computed in polynomial time in the input size and also the number of nodes is linear in the input size. Thus, the ETH bound follows.

Correctness: ($\Rightarrow$) Let $S \subseteq V(G)$ of size κ be an independent set. To create polytree $T^\star$, we add the parent set $D_q := V_2 \cup \{p\}$ to T while removing the current parent set $\{v_1\}$ of q and further arcs of T to ensure acyclicity. More specifically, polytree $T^\star$ consists of the following arcs:

- each arc of $D_q = \{(x, q) : x \in V_2 \cup \{p\}\}$,
- arc (v_2, v_1) if $v_1 \notin S$, and
- arc (v_1, p) if $v_1 \in S$.

We now evaluate the score of $T^\star$: Since S is an independent set of size κ, we have $f_p(S) = \alpha\kappa$. Also, we have $f_q(V_2 \cup \{p\}) = W \cdot (n+1) = 3n^2 + 5n - m + \kappa + 2$. Moreover, we have $f_{v_1}(v_2) = \beta$ for each $v_1 \in V \setminus S$. Consequently, we have $s(T^\star) = \alpha\kappa + W(n+1) + \beta(n-\kappa) = \beta n + W(n+1) + \kappa = W(n+1) + an - m - \tau = s(T) - 1 > s(T)$.

($\Leftarrow$) Let $T^\star$ be a solution of the BEST IMPROVED POLYTREE instance, that is, $s(T^\star) \geq \beta \cdot n + \alpha \cdot n - m + c + 1 = 3n^3 + 11n^2 + 11n + 2 - m(n+1) - \kappa n$. Clearly, since every vertex except q already has its unique maximal parent set, in $T^\star$, vertex q received a new parent set $D_q \subseteq V_2 \cup \{p, v_1\}$ and let $B := D_q \cap (V_2 \cup \{p\})$. Moreover, let X be the arc set which got removed by the algorithm to to acyclicity. Initially, we argue that q needs to have parent set $V_2 \cup \{p\}$. Assume towards, a contradiction that $v_1 \in D_q$. Initially, observe that $|X| \geq |B|$ since T is a spanning tree and every arc from B to q increases the number of arcs by one. With the following claim we bound the score lost by the removal of the arcs from X which occurs at the vertices distinct from q. Note that this bound is also correct if $v_1 \notin D_q$.

Claim B.8. The total score lost at the vertices distinct from q is at least $(\alpha - 3) \cdot |X|$.

Proof of Claim. Clearly, X contains only arcs of the form (V^2, v^1) (type 1) or (v^1, p) (type 2). Note that each deleted arc of type 1 drops the score by $\beta = \alpha - 1$. Moreover, let $S \subseteq V_1$ be the parent set of p in $T^\star$. Then, the arcs (z, p) for $z \in V_1 \setminus S$ are of type 2, and these are $|V_1 \setminus S|$ many. The total score loss at vertex p is thus $f_p(V_1) - f_p(S) = \alpha|V_1 \setminus S| - (|E(G)| - |E(G[S])|) \geq (\alpha - 3)|V_1 \setminus S|$ since every edge counted in $|E(G)| - |E(G[S])|$ has an endpoint in $V_1 \setminus S$ and G is subcubic; so there are at most $3|V_1 \setminus S|$ of them. Thus, the statement follows. ■

Now, we use Claim B.8 to verify that $s(T^\star) \leq s(T)$. Clearly, q is the unique vertex which received a larger score in $T^\star$ compared to T. More precisely, we have $f_q(\{v_1\} \cup B) - f_q(\{v_1\}) = \rho|B|$. By Claim B.8, the score loss is at least $(\alpha - 3) \cdot |X| \geq (\alpha - 3)|B|$. Since $\tau \leq 3n^2 + 5n + 1 < 3n^2 + 5n + 2 = (\alpha - 3)(n+1)$, we obtain that $\rho = \frac{\tau}{n+1} < \alpha - 3$. Thus, $s(T^\star) \leq s(T)$ and equality is only possible if $B = \emptyset$, that is, if $T^\star = T$. This is a contradiction to our assumption that $s(T^\star) \geq s(T)$. Hence, we have $v_1 \notin D_q$.

In other words, we have $D_q \subseteq V_2 \cup \{p\}$. Let $k := |D_q|$. Since q needs to receive a parent set with strictly larger score in $T^\star$ than in T we obtain the following: We now have $f_q(D_q) - f_q(\{v_1\}) = W \cdot k - c \leq W \cdot n - c = Wn - (W \cdot n + 1) = -1 < 0$ for each $k \leq n$. Consequently, we need to have $k = n + 1$, that is, $D_q = V_2 \cup \{p\}$. Note that this is consistent with the fact that $c < W(n+1) = (\tau + 1)(n+1)$ since $\tau = \alpha n - m - \kappa + 1 \geq 3n^2 = 5n/2 + 1 > 0$.

Note that for each $v \in V$ the arcs (v^2, q) and (p, q) yield the cycle $q - v^2 - v^1 - p - q$. Since $D_q = V_2 \cup \{p\}$, we know that exactly one arc of (v^2, q) and (p, q) is also contained in $T^\star$ while the other got removed to to acyclicity (if for one $v \in V(G)$ both arcs (v_1, p) and (v_2, v_1) are not part of p, we can safely add arc (v_2, v_1) without violating acyclicity). Let $S \subseteq V_1$ be the parent set of p in $T^\star$. Consequently, $T^\star$ has score $s(T^\star) = \beta(n - |S|) + \alpha|S| - |E(G[S])| + W(n+1)$. Due to $\alpha = \beta + 1$, we obtain that $s(T^\star) - s(T) = (n\beta + (|S| - |E(G[S])|) + W(n+1)) - (n\beta + (an - m) + c) = (|S| - |E(G[S])|) + (W(n+1) - c) - (an - m) = (|S| - |E(G[S])|) - \kappa - 1$. Hence, we have $s(T^\star) - s(T) \geq 1$ if and only if $|S| \geq |E(G[S])| + \kappa$. We now argue that S needs to contain an independent set I of size at least κ; once we have achieved this, we have finished the proof. We now iteratively remove edges and one incident vertex as follows: Set $S_0 := S$ and repeat the following while $G[S_i]$ contains at least one edge $\{x, y\}$: set $S_{i+1} = S_i \setminus \{x\}$. Clearly, $|S_{i+1}| = |S_i| - 1$ and $|E(G[S_{i+1}])| \leq |E(G[S_i])| - 1$. Thus, $|S_{i+1}| - |E(G[S_{i+1}])| \geq |S_i| - |E(G[S_i])|$. Note that this process stops after at most $|E(G[S])|$ steps and returns an independent set I. Consequently, graph G contains an independent set I of size at least κ. □

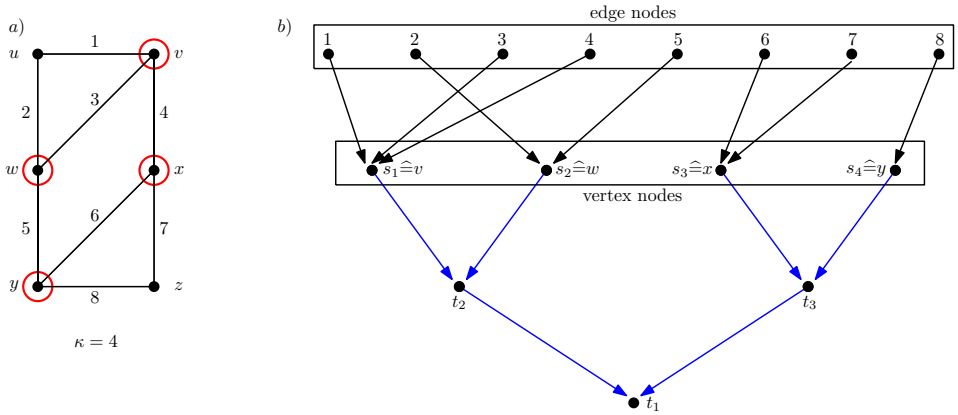


Figure 5: Visualization of the reduction behind Theorem 3.4. Part a) shows a CUBIC VERTEX COVER instance with $\kappa = 4$ (the property of being cubic is dropped) and the red vertices correspond to a vertex cover of size 4. Part b) shows a solution to the equivalent instance of $\text{PT}^{\text{sm}}_{\leq}$. There, each blue arc yields score n^2 and each black arc yields score 1. The shown polytree thus has a score of $24 \cdot n^2 + 32 = t$.

B.8 PROOF OF THEOREM 3.4

Theorem 3.4 (★). *For each $k \geq 4$, $\text{PT}^{\text{sm}}_{\leq}$ is NP-hard and cannot be solved in $2^{o(n)}$ time unless the ETH fails.*

Proof. We provide a reduction from CUBIC VERTEX COVER. There, the input consists of a graph G where each vertex has degree 3, and an integer κ. The goal is to find a *vertex cover* $S \subseteq V(G)$, that is, every edge of G has at least one endpoint in S, of size at most κ. CUBIC VERTEX COVER is NP-hard (Garey & Johnson, 1979). Moreover, CUBIC VERTEX COVER cannot be solved in $2^{o(n)}$ time unless the ETH fails (Impagliazzo et al., 2001). Without loss of generality, we assume that κ is a power of 2, and that $n \geq 12$.

Construction. A visualization is provided in Figure 5. Our node set consists of three parts: (1) For each edge $e \in E(G)$, we create an *edge node* e. (2) We create κ *vertex nodes* $v_1, \ldots, v_\kappa$. (3) We create $(\kappa - 1)$ *tree nodes* $t_1, \ldots, t_{\kappa-1}$. The idea is that the parent sets of the tree nodes induce a binary tree on the tree nodes and the vertex nodes. We guarantee this by assigning these parent sets a huge score. Moreover, our goal is that in a polytree with highest score, each vertex node can pick a subset S_v of the incident edges $\Gamma(v)$ of any vertex $v \in V(G)$ as a parent set which yields score $4|S_v|$. Technically, a node can pick edge sets which are not incident to a single node but this will not yield a larger score. The binary tree on the vertex nodes and tree nodes ensures that each edge node is part of at most one parent set. The score bound t is set such that each edge node has to be contained in at least one parent set and such that each vertex node is only allowed to pick edges incident to a specific vertex of G. This then ensures that the parent sets of the vertex nodes correspond to a vertex cover S of $V(G)$.

We now describe the scores of the maximal parent sets (and of some of their subsets).

- Each edge node only has the empty parent set which gives score 0.
- By $\Gamma(v)$ we denote the set of incident edges with vertex v. We say an edge set E' is a *star* if $E' \subseteq \Gamma(v)$ for some $v \in V$, that is, if all edges of E' are incident to a common vertex v. Otherwise, we say that E' is *star-free*.

  The score functions of all v_i are identical and thus in the following we only describe f_{v-i}.

  We begin by describing the maximal parent sets:

$$\mathcal{F}_{v_i} := \{\Gamma(v) : v \in V(G)\} \cup \{S \subseteq E(G) : |S| = 4\}.$$

Intuitively, the maximal parent sets are all 3-stars and all sets of 4 edges (which cannot be a 4-star since G is cubic). Clearly, we have $k = 4$. Note that we require G to be cubic instead of subcubic to have an easier description of $\mathcal{F}_{v_i}$.

For any edge set $R \subseteq E$, we set

$$f_{v_i}(R) := \begin{cases} 4\,|R|, & R \text{ a star}, \\ \min\big(12,\ 3|R|+1\big), & \text{otherwise.} \end{cases}$$

Note that $f_{v_i}(R) = 4|R|$ if and only if R is a star in G. This property is crucial for the correctness.

We now argue that f_{v_i} is monotone and submodular:

Set $\mu(Q, e) := f_{v_i}(Q \cup \{e\}) - f_{v_i}(Q)$ for $e \notin Q$. Observe that μ can only have values $4, 3, 2, 0$:

$$\begin{array}{ll} \mu = 4: & Q = \emptyset;\ \ |Q| \leq 2 \text{ and } Q \cup \{e\} \text{ a star}, \\ \mu = 3: & |Q| = 1 \text{ and } Q \cup \{e\} \text{ not a star};\ \ Q \text{ a 2-matching}, \\ \mu = 2: & Q \text{ a 2-star and } Q \cup \{e\} \text{ not a star};\ \ |Q| = 3,\ Q \text{ not a star}, \\ \mu = 0: & Q \text{ a 3-star, or } |Q| \geq 4. \end{array}$$

For $|Q| = 2$ note that $Q \cup \{e\}$ can be a star if and only if Q is a 2-star. All values are ≥ 0, so f_{v_i} is monotone.

For submodularity let $Q \subseteq Q'$ and $e \notin Q'$. We now check $\mu(Q, e) \geq \mu(Q', e)$ by the value of $\mu(Q', e)$.

- $\mu(Q', e) = 0$: There is nothing to prove, since $\mu \geq 0$ throughout.
- $\mu(Q', e) = 4$: Hence, $Q' \cup \{e\}$ is a star and $|Q'| \leq 2$. Then $Q \cup \{e\} \subseteq Q' \cup \{e\}$ is a star as well, and $|Q| \leq |Q'| \leq 2$, so we obtain $\mu(Q, e) = 4 = \mu(Q', e)$.
- $\mu(Q', e) = 3$: Hence Q' is either a single edge such that $Q' \cup \{e\}$ is not a star, or Q' is a 2-matching. In particular we have $|Q'| \leq 2$. The only possible cases for Q are:
  - $Q = \emptyset$. Then $Q \cup \{e\} = \{e\}$ is a 1-star, and thus $\mu(Q, e) = 4$.
  - $Q = Q'$. Then $\mu(Q, e) = \mu(Q', e) = 3$.
  - Q a single edge with $Q \subsetneq Q'$. This occurs only when Q' is a 2-matching (if Q' is a single edge, the only proper subset is $\emptyset$). Since $|Q| = 1$, we have $\mu(Q, e) = 4$ if $Q \cup \{e\}$ is a star, and $\mu(Q, e) = 3$ if it is not.

  These cases exhaust $\{Q : Q \subseteq Q'\}$, and in each of them $\mu(Q, e) \geq 3 = \mu(Q', e)$.
- $\mu(Q', e) = 2$: Here it suffices to show that $\mu(Q, e) \geq 2$. By the above, $\mu(Q, e) < 2$ implies $\mu(Q, e) = 0$ which only happens when Q is a 3-star or $|Q| \geq 4$. Hence Q' is either a 2-star with $Q' \cup \{e\}$ not a star, or a non-star with $|Q'| = 3$.
  - Q' a 2-star. Then $|Q| \leq |Q'| = 2$, so Q is neither a 3-star nor of size ≥ 4, and $\mu(Q, e) \geq 2$.
  - $|Q'| = 3$ and Q' not a star. Either $Q = Q'$, and then $\mu(Q, e) = 2$, or $|Q| \leq 2$, and then $\mu(Q, e) \geq 2$ as in the previous line. In neither case can Q be a 3-star, because the only 3-element subset of Q' is Q' itself, which is not a star.

  So $\mu(Q, e) \geq 2 = \mu(Q', e)$.

Hence, we have verified that f_{v_i} is submodular.

- For the tree nodes, we distinguish whether $i < \kappa/2$ or not. For $i < \kappa/2$, we set $f_{t_i}(X) = 4n^2 \cdot |X|$, for any $X \subseteq \{t_{2i}, t_{2i+1}\}$ and each other parent set gives score 0.

  For i with $\kappa/2 \leq i \leq \kappa - 1$, we set $f_{t_i} = 4n^2 \cdot |X|$, for any $X \subseteq \{v_{2(i-\kappa/2+1)-1}, v_{2(i-\kappa/2+1)}\}$ and each other parent set gives score 0.

  Clearly, in both cases f_{t_i} is submodular.

Finally, we set the score bound $t := (2\kappa - 2) \cdot n^2 + 4m$, where m is the number of edges of G. Clearly, the reduction can be carried out in polynomial time. Moreover, since the number of nodes is

linear in the size of the CUBIC VERTEX COVER, the ETH bound follows. The maximum parent set size of 4 is achieved by any vertex node.

Correctness. We show that G has a vertex cover S of size κ if and only if there is a polytree on N with score at least t.

($\Rightarrow$) Let S be a vertex cover of size at most κ of G and let $s_1, \ldots, s_\kappa$ be an arbitrary but fixed ordering of S. We now construct a polytree on N which has score t.

Each tree node t_i selects its unique parent set of size 2 which yields score $8n^2$. Thus, the score of all tree nodes together is $(8\kappa - 8) \cdot n^2$. Now the idea is to assign vertex node v_i the edges covered by vertex s_i of the vertex cover. We need to be, however, a bit careful since some edges e may have both endpoints in S and then e can only be part of one parent set of the vertex nodes to preserve acyclicity. Hence, we set S_i to be the edges covered by $\{s_1, \ldots, s_i\}$. Note that $S_\kappa = E(G)$ since $S = \{s_1, \ldots, s_\kappa\}$ is a vertex cover of G. We now assign parent set $\Gamma(s_i) \setminus S_{i-1}$ to vertex node v_i. Note that if S contains less than κ vertices, some vertex nodes just pick the empty parent set. This ensures that in the underlying graph of the polytree each edge node has degree 1. Thus, the total score of all vertex nodes is $4m$ (since the parent set of each vertex node is a star) and thus the total score of all nodes is $(8\kappa - 8)n^2 + 4m = t$.

Moreover, we indeed face a polytree: Each edge node has degree 1. After the removal of all edge nodes, all vertex nodes have degree 1, and then the parent sets of all tree nodes induce a binary tree.

($\Leftarrow$) Let T be a polytree on N with score at least $t = (8\kappa - 8)n^2 + 4m$. First, we argue that each tree node has to select its unique parent set of size 2. Note that each edge node can only pick the empty parent set, and that the maximal score of any vertex node is 12. Consequently, all edge nodes and vertex nodes together yield a score of at most $\max(4m, 12\kappa) < 4n^2$. By the above assumption the total score of all tree nodes is at most $(8\kappa - 9)n^2$, polytree T cannot have score t in that case. Hence, it is safe to assume that each tree node has its unique parent set of size 2. Consequently, all tree nodes together contribute $(8\kappa - 8)n^2$ to the score of T.

Recall that any edge node can only pick the empty parent set. Thus, all vertex nodes together need to contribute $4m$ to t. By definition, we have $D_{v_i} \subseteq E(G)$. Observe that the sets D_{v_i} are pairwise disjoint: If there exist an edge $e \in D_{v_i} \cap D_{v_j}$, then the two arcs (e, v_i) and (e, v_j) together with the path between v_i and v_j implies by the tree nodes yields a cycle in the underlying graph. Consequently, we have $\sum_{i \in [\kappa]} |D_{v_i}| \leq m$. By the fact that $f_{v_i}(E') \leq 4|E'|$ for each $E' \subseteq E$ where equality only holds for stars, we obtain that

$$4m \leq \sum_{i=1}^{\kappa} f_{v_i}(D_{v_i}) \leq \sum_{i=1}^{\kappa} 4|D_{v_i}| \leq 4m.$$

Consequently, equality holds throughout and thus $f_{v_i}(D_{v_i}) = 4|D_{v_i}|$ for each v_i. Hence, each D_{v_i} is a star and the $(D_{v_i})_{i \in [\kappa]}$ form a partition of $E(G)$. For each non-empty D_{v_i} let s_i be a vertex with $D_{v_i} \subseteq \Gamma(si)$. Note that s_i is unique if $|D_{v_i}| \geq 2$ and that either endpoint is viable if $|D_{v_i}| = 1$. Then $S := s_i : D_{v_i} \neq \emptyset$ has size at most κ and covers every edge of G, because the D_{v_i} are a partition of $E(G)$. Thus, S is a vertex cover of size κ for G. □

B.9 FURTHER DETAILS FOR METRICS FOR DISTANCE TO SUBMODULARITY FUNCTIONS

Further details for the submodularity ratio. which is equivalent to $\sum_{u \in T} \mathsf{mG}(u \mid S) \geq \gamma \cdot \mathsf{mG}(T \mid S)$ whenever $\mathsf{mG}(T \mid S) \geq 0$. Function f is submodular if and only if $\gamma_{U',k} \geq 1$ for all U' and all k (Das & Kempe, 2018, Proposition 3). Consequently, if f is not submodular, we have $\gamma_{U',k} < 1$.

Further details for the α-curvature. Note that $\alpha = 0$ yields that f is additive and, in contrast, $\alpha = 1$ poses no restrictions on (the monotone increasing) f. If $\alpha < 1$ then in particular α measures the violation of the marginal-gains characterization of submodularity (when we put $A \supseteq B$).

Further details for the marginal gain violation. Observe that in the element mG-violation we only consider sets C of size one. Consequently, $\epsilon^{\text{ele}}_{\mathsf{mG}} \leq \epsilon_{\mathsf{mG}}$. Moreover, observe that $\epsilon_{\mathsf{mG}} = 0 = \epsilon^{\text{ele}}_{\mathsf{mG}}$ if and only if f is submodular. Intuitively, ϵ_{mG} and $\epsilon^{\text{ele}}_{\mathsf{mG}}$ measure the distance of f being submodular.

Note that element mG-violation $\epsilon^{\text{ele}}_{\mathsf{mG}}$ is closely related to the α-curvature: while in $\epsilon^{\text{ele}}_{\mathsf{mG}}$ we require that $B \subseteq A$, in the α-curvature we do not impose that restriction.

B.10 Proof of Theorem 3.5

Theorem 3.5 (★). *1.* $\mathrm{PT}_{\leq}^{(\alpha,\gamma)}$ *and* $\mathrm{PT}^{(\alpha,\gamma)}$ *admit a* $(1 + 1/(\gamma \cdot (1-\alpha)))$*-approx. algorithm.*
2. $\mathrm{PT}_{\leq}^{\mathsf{mG}}$ *and* $\mathrm{PT}^{\mathsf{mG}}$ *admit a* $(2 + \epsilon_{\mathsf{mG}})$*-approximation algorithm.*
3. $\mathrm{PT}_{\leq}^{\mathsf{mG},\gamma}$ *and* $\mathrm{PT}^{\mathsf{mG},\gamma}$ *admit a* $(1 + (1 + \epsilon_{\mathsf{mG}}^{ele})/\gamma)$*-approximation algorithm.*

Proof of Theorem 3.5. Recall that in any case we assume f to be monotone and that this assumption is used in Equation (14) in the proof of Theorem 3.1 and to argue that $|\,\mathsf{fes}(Z)| \leq |Z|$ for each acyclic arc set Z. Moreover, the fact that the scoring function of each parent set is submodular is only used in Equation (10) in the proof of Theorem 3.1 and in the subsequent Equations (11), (12) and (15). All other arguments on the existence of the exchange mapping f do not rely on the assumption that the scoring function s is submodular. Hence, in the following, we revisit all arguments starting from Equation (10) and adapt them to the three variants.

1. We begin with $\mathrm{PT}_{\leq}^{\mathsf{mG}}$ *and* $\mathrm{PT}^{\mathsf{mG}}$. In other words, we consider monotone increasing functions with marginal gain violation ϵ_{mG}. Recall that in Equation (10) $\mathsf{mG}(Q_w^{v_i} \mid (D_A^w \setminus Q_w^{v_i}))$ is upper bounded by $\mathsf{mG}(Q_w^{v_i} \mid \bigcup_{j=1}^{i-1} Q_w^{v_j})$. Note that $\bigcup_{j=1}^{i-1} Q_w^{v_j} \subseteq D_A^w \setminus Q_w^{v_i}$ and that $Q_w^{v_i} \cap (D_A^w \setminus Q_w^{v_i}) = \emptyset$. Thus, we can exploit the marginal gain violation ϵ_{mG} and obtain:

$$\mathsf{mG}(Q_w^{v_i} \mid (D_A^w \setminus Q_w^{v_i})) \leq (1 + \epsilon_{\mathsf{mG}}) \cdot \mathsf{mG}(Q_w^{v_i} \mid \bigcup_{j=1}^{i-1} Q_w^{v_j}). \tag{19}$$

Next, the proof works similar to the proof of Theorem 3.1. Here, we only provide the details until the equivalent equation to Equation (14).

By the definition of marginal gains, we have

$$s(\bigcup_{j=1}^{i-1} Q_w^{v_j}) + \mathsf{mG}(Q_w^{v_i} \mid \bigcup_{j=1}^{i-1} Q_w^{v_j}) = s(\bigcup_{j=1}^{i} Q_w^{v_j}), \tag{20}$$

and thus we obtain

$$\sum_{v \in N \setminus \{w\}} \mathsf{mG}(Q_w^v \mid (D_A^w \setminus Q_w^v)) \tag{21}$$

$$\leq \sum_{v \in N \setminus \{w\}} (1 + \epsilon_{\mathsf{mG}}) \cdot \mathsf{mG}(Q_w^{v_i} \mid \bigcup_{j=1}^{i-1} Q_w^{v_j}) \tag{22}$$

$$\leq (1 + \epsilon_{\mathsf{mG}}) \cdot \sum_{v \in N \setminus \{w\}} \mathsf{mG}(Q_w^{v_i} \mid \bigcup_{j=1}^{i-1} Q_w^{v_j}) \tag{23}$$

$$\leq (1 + \epsilon_{\mathsf{mG}}) \cdot s(D_A^w). \tag{24}$$

Consequently, we obtain

$$\sum_{w \in N} \sum_{v \in N \setminus \{w\}} \mathsf{mG}(Q_w^v \mid (D_A^w \setminus Q_w^v)) \tag{25}$$

$$\leq \sum_{w \in N} (1 + \epsilon_{\mathsf{mG}}) \cdot s(D_A^w) = (1 + \epsilon_{\mathsf{mG}}) \cdot s(A). \tag{26}$$

In Equation (26), we exploit that f is monotone. Hence, we have proven that A yields an approximation factor of $1 + (1 + \epsilon_{\mathsf{mG}}) = 2 + \epsilon_{\mathsf{mG}}$ for S.

2. We continue with $\mathrm{PT}_{\leq}^{(\alpha,\gamma)}$ *and* $\mathrm{PT}^{(\alpha,\gamma)}$. In other words, we consider monotone increasing functions with curvature $0 < \alpha < 1$ and submodularity ratio $0 < \gamma < 1$.

Recall Equation (10), where $v_1, \ldots v_{n-1}$ is an arbitrary but fixed ordering of $N \setminus \{w\}$.

$$\mathsf{mG}(Q_w^{v_i} \mid (D_A^w \setminus Q_w^{v_i})) \leq \mathsf{mG}(Q_w^{v_i} \mid \bigcup_{j=1}^{i-1} Q_w^{v_j}), \tag{27}$$

and that afterwards Equation (10) was summed over all nodes $v \in N \setminus w$ to obtain Equation (12)

$$\sum_{v \in N \setminus \{w\}} \mathsf{mG}(Q_w^v \mid (D_A^w \setminus Q_w^v)) = s(\bigcup_{j=1}^{n-1} Q_w^{v_j}) \tag{28}$$

$$\leq s(D_A^w). \tag{29}$$

which then was used to argue that the sum over all $w \in N$ then yields the score $s(A)$ of the approximate solution (Equation (15)). Again, in the last inequality we exploited that f is monotone.

Now, our goal is to obtain a similar bound for $\mathsf{mG}(Q_w^{v_i} \mid (D_A^w \setminus Q_w^{v_i}))$ where we take γ and α into account. Since the α-curvature is only defined for single elements, we cannot adapt Equation (10) directly, but rather have to go from sets of elements to single elements first. To do so, we exploit the submodularity ratio γ, as follows:

$$\mathsf{mG}(Q_w^{v_i} \mid (D_A^w \setminus Q_w^{v_i})) \leq 1/\gamma \cdot \sum_{u \in Q_w^{v_i}} \mathsf{mG}(u \mid (D_A^w \setminus Q_w^{v_i})). \tag{30}$$

We now aim to upper bound

$$\sum_{v \in N \setminus \{w\}} \mathsf{mG}(Q_w^{v_i} \mid (D_A^w \setminus Q_w^{v_i})) \tag{31}$$

$$\leq \sum_{v_i \in N \setminus \{w\}} 1/\gamma \cdot (\sum_{u \in Q_w^{v_i}} \mathsf{mG}(u \mid (D_A^w \setminus Q_w^{v_i}))) \tag{32}$$

$$= 1/\gamma \cdot \sum_{v_i \in N \setminus \{w\}} \sum_{u \in Q_w^{v_i}} \mathsf{mG}(u \mid (D_A^w \setminus Q_w^{v_i})). \tag{33}$$

Now let $u_1^{v_i}, u_2^{v_i} \ldots, u_\ell^{v_i}$ be an arbitrary but fixed ordering of the arcs of $Q_w^{v_i}$. Moreover, let $U_x^{v_i} := \{u_j^{v_i} : j < x\} \subsetneq Q_w^{v_i}$. We now extend the $u_x^{v_i}$'s to an ordering of D_A^w as follows. For $u_x^{v_i}$, set $U_x := U_x^{v_i} \cup \bigcup_{j<i} Q_w^{v_j}$, that is, the set U_x additionally to $U_x^{v_i}$ also contains all 'previous' sets $Q_w^{v_j}$.

Recall that each local score function s_v is monotone increasing and has curvature α. Since for $u_x^{v_i} \in Q_w^{v_i}$, we have $u_x^{v_i} \notin D_A^w \setminus Q_w^{v_i}$ and $u_x^{v_i} \notin U_x$, we obtain that

$$\mathsf{mG}(u_x^{v_i} \mid (D_A^w \setminus Q_w^{v_i})) \leq 1/(1-\alpha) \cdot \mathsf{mG}(u_x^{v_i} \mid U_x). \tag{34}$$

Consequently,

$$1/\gamma \cdot \sum_{v \in N \setminus \{w\}} \sum_{u \in Q_w^{v_i}} \mathsf{mG}(u \mid (D_A^w \setminus Q_w^{v_i})) \tag{35}$$

$$\leq 1/\gamma \cdot 1/(1-\alpha) \cdot \sum_{v_i \in N \setminus \{w\}} \sum_{u_x^{v_i} \in Q_w^{v_i}} \mathsf{mG}(u_x^{v_i} \mid U_x) \tag{36}$$

$$\leq 1/((1-\alpha)\cdot\gamma) s(D_A^w), \tag{37}$$

where the last inequality is due to the definition of the marginal gain $\mathsf{mG}(\cdot)$. Again, the last inequality can be strict since the $Q_w^{v_i}$ do not need to form a partition of D_A^w.

From now on the proof is analogous to the proof of Theorem 3.1, where we now additionally have a factor $1/((1-\alpha)\cdot\gamma)$ instead of 1.

3. Finally, we consider $\mathrm{PT}_{\leq}^{\mathsf{mG},\gamma}$ *and* $\mathrm{PT}^{\mathsf{mG},\gamma}$. In other words, we consider functions with element-wise marginal gain violation $\epsilon_{\mathsf{mG}}^{\mathsf{ele}}$ and submodularity ratio $0 < \gamma < 1$.

The only difference to 2. (and that is where the α-curvature is used) is in Equation (34). For $\mathrm{PT}_{\leq}^{\mathsf{mG},\gamma}$ and $\mathrm{PT}^{\mathsf{mG},\gamma}$ we can bound $\mathsf{mG}(u_x^{v_i} \mid (D_A^w \setminus Q_w^{v_i}))$ by $\mathsf{mG}(u_x^{v_i} \mid U_x)$ as follows: Since $U_x \subseteq D_A^w \setminus Q_w^{v_i}$ we can exploit the element-wise marginal gain violation $\epsilon_{\mathsf{mG}}^{\mathsf{ele}}$ as follows:

$$\mathsf{mG}(u_x^{v_i} \mid (D_A^w \setminus Q_w^{v_i})) \leq (1 + \epsilon_{\mathsf{mG}}^{\mathsf{ele}}) \cdot \mathsf{mG}(u_x^{v_i} \mid U_x). \tag{38}$$

Afterwards, the proof is completely analogous to 2. and thus the result follows. □

B.11 Proof of Theorem 3.6

Theorem 3.6 (★). $\mathrm{PT}^{\mathrm{set}}_{\leq}$ *admits a* $\big(2 + 2k \cdot \epsilon_{set}/(1 + \epsilon_{set})\big)$*-approximation algorithm. Recall that* k *is the maximum parent-set size.*

Proof of Theorem 3.6. As in the proof of Theorem 3.5, everything before Equation (10) of the proof of Theorem 3.1 carries over; we only replace the use of submodularity at Equation (10) by an inequality exploiting ϵ_{set}. Again, the fact that f is monotone is exploited only once in the last inequality when we go from a subset to a parent set of an optimal solution. Now the goal is to upper bound $\mathsf{mG}(Q_w^{v_i} \mid D_A^w \setminus Q_w^{v_i})$ by $\mathsf{mG}(Q_w^{v_i} \mid \bigcup_{j<i} Q_w^{v_j})$, incorporating the set-based violation ϵ_{set}.

Fix w and an ordering $v_1, \ldots, v_{n-1}$ of $N \setminus \{w\}$. Observe that $\bigcup_{j=1}^{i-1} Q_w^{v_j} \subseteq D_A^w \setminus Q_w^{v_i}$ and that $Q_w^{v_i} \cap (D_A^w \setminus Q_w^{v_i}) = \emptyset$. We now apply the set-based violation ϵ_{set} with the sets

$$S' \coloneqq D_A^w \setminus Q_w^{v_i}, \qquad T' \coloneqq \big(\bigcup_{j=1}^{i-1} Q_w^{v_j}\big) \cup Q_w^{v_i}.$$

This yields $S' \cup T' = D_A^w$ and $S' \cap T' = \bigcup_{j=1}^{i-1} Q_w^{v_j}$.

For convenience we write:

- $a \coloneqq s(D_A^w) \geq 0$,
- $\alpha_i \coloneqq \mathsf{mG}(Q_w^{v_i} \mid D_A^w \setminus Q_w^{v_i})$,
- $\beta_i \coloneqq \mathsf{mG}(Q_w^{v_i} \mid \bigcup_{j=1}^{i-1} Q_w^{v_j})$, and
- $b_i \coloneqq s(\bigcup_{j=1}^{i-1} Q_w^{v_j})$.

Observe that

- $s(S') = a - \alpha_i$,
- $s(T') = b_i + \beta_i$,
- $s(S' \cup T') = a$, and
- $s(S' \cap T') = b_i$.

Thus from the definition $(1 + \epsilon_{\text{set}})(s(S') + s(T')) \geq s(S' \cup T') + s(S' \cap T')$ of the set-based violation, we obtain

$$(1 + \epsilon_{\text{set}})\big((a - \alpha_i) + (b_i + \beta_i)\big) \;\geq\; a + b_i. \tag{39}$$

Solving (39) for α_i yields

$$\alpha_i \;\leq\; \beta_i + \eta \cdot (a + b_i), \qquad \text{where} \quad \eta \coloneqq \tfrac{\epsilon_{\text{set}}}{1+\epsilon_{\text{set}}}. \tag{40}$$

We now re-investigate the vertices $v_1, \ldots, v_{n-1}$ of $N \setminus \{w\}$. Clearly, for many indices i, we have $Q_w^{v_i} = \emptyset$. Thus, by $I \subseteq [n-1]$ we denote the set of all indices such that $Q_w^{v_i} \neq \emptyset$. Moreover, recall that the $Q_w^{v_i}$ are disjoint but they do not necessarily form a partition of D_A^w and that $|D_A^w| \leq k$. Consequently, we have $|I| \leq k$.

We now sum over Equation (40) on each index in I.

$$\sum_{i \in I} \alpha_i \leq \sum_{i \in I} \beta_i + \eta \sum_{i \in I} (a + b_i) \tag{41}$$

Observe that by the definition of the marginal gains, we have $\sum_{i \in I} \beta_i = a$. Moreover, observe that $b_i = s(S' \cap T') = s(\bigcup_{j=1}^{i-1} Q_w^{v_j}) \leq s(D_A^w) = a$. Consequently, we obtain

Table 2: Approximation factor $\rho_\emptyset$ on the original (unshifted) BDeu scores guaranteed by outputting the empty polytree, and the approximation factor ρ_{orig} on the original (unshifted) BDeu scores implied by our algorithm.

	adult	car	chess-KRK	dna	mushroom	nursery	solar-flare	vote
$\rho_\emptyset$	1.72	1.11	1.14	1.04	3.27	1.23	1.66	1.41
ρ_{orig}	1.50	1.09	1.12	1.03	3.17	1.19	1.43	1.26

$$\sum_{i \in I} \alpha_i \leq a + \eta \sum_{i \in I} (a + a) \leq (1 + 2\eta k)a. \tag{42}$$

In other words, Equation (42) yields

$$\sum_{v \in N \setminus \{w\}} \mathsf{mG}(Q_w^v \mid D_A^w \setminus Q_w^v) \leq (1 + 2\eta k) \cdot s(D_A^w). \tag{43}$$

Now, summing Equation (43) over all $w \in N$ yields

$$\sum_{w \in N} \sum_{v \in N \setminus \{w\}} \mathsf{mG}(Q_w^v \mid D_A^w \setminus Q_w^v) \leq \sum_{w \in N} (1 + 2\eta k) \cdot s(D_A^w) = (1 + 2\eta k) \cdot s(A). \tag{44}$$

Consequently, we obtain a $(2 + 2\eta k)$-approximation if we assume set-based violation ϵ_{set}. In other words, the approximation guarantee is $\big(2 + 2k \cdot \epsilon_{\text{set}}/(1 + \epsilon_{\text{set}})\big)$. □

Recall that we used Lemma B.4 to argue that the assumption of f being monotone is not necessary to proveTheorem 3.1. Moreover, monotonicity was used to argue that $|\,\mathsf{fes}(Z)| \leq |Z|$ for each acyclic arc set Z. In the latter case, we already argued that monotonicity is not required if `LS-PT` is restricted to add reasonable parent sets. Now observe that the identical argument in Lemma B.4 also applies to the proofs of Theorems 3.5 and 3.6, that is, by exploiting Lemma B.4 one can drop the assumption of f being monotone.

C Additional Material for Section 4

C.1 Unshifted approximation factors implied by our algorithm

Implied approximation factors for the original, unshifted scores are given in Table 2: Theorem 3.6 provides the factor $\rho := 2 + 2k\,\epsilon_{\text{set}}/(1 + \epsilon_{\text{set}})$ for the shifted scores $s_{\text{shift}}(D) = s_{\text{orig}}(D) + C$, where $C := \sum_v \text{shift}_v$ is the total shift introduced above; rearranging the shifted-score guarantee for an algorithm output A and an optimal polytree $S^\star$ gives (see below for the full derivation)

$$\frac{|s_{\text{orig}}(A)|}{|s_{\text{orig}}(S^\star)|} \;\leq\; \frac{1}{\rho} + \Big(1 - \tfrac{1}{\rho}\Big)\frac{C}{|s_{\text{orig}}(S^\star)|} \;\leq\; \underbrace{\frac{1}{\rho} + \Big(1 - \tfrac{1}{\rho}\Big)\frac{C}{\big|\sum_v \max_P f_{v,\text{orig}}(P)\big|}}_{=:\,\rho_{\text{orig}}}.$$

The second inequality uses $|s_{\text{orig}}(S^\star)| \geq \big|\sum_v \max_P f_{v,\text{orig}}(P)\big|$: the per-node maxima ignore acyclicity, so $s_{\text{orig}}(S^\star) \leq \sum_v \max_P f_{v,\text{orig}}(P)$, and on each of our datasets the right-hand side is negative, so taking absolute values reverses the inequality. Even though ρ ranges between roughly 2.7 and 22, ρ_{orig} stays much closer to 1 throughout (last row of Table 2). For calibration, Table 2 also

lists $\rho_\emptyset := C \,/\, \left|\sum_v \max_P f_{v,\mathrm{orig}}(P)\right|$, the factor that the same derivation guarantees for the trivial algorithm that returns the empty polytree $A_\emptyset$; indeed $C = |s_{\mathrm{orig}}(A_\emptyset)|$, because the pruned scores are monotone and we shift each node by its minimum. By definition we have $\rho_{\mathrm{orig}} = 1/\rho + (1 - 1/\rho)\,\rho_\emptyset$, a convex combination of 1 and $\rho_\emptyset$; in particular $\rho_{\mathrm{orig}} \in [1, \rho_\emptyset]$ regardless of ρ.

Set $\rho := 2 + 2k\,\epsilon_{\text{set}}/(1 + \epsilon_{\text{set}})$, let $C := \sum_v \mathrm{shift}_v$ denote the total introduced shift, and write $s_{\mathrm{shift}}(D) = s_{\mathrm{orig}}(D) + C$ for the shifted score of any polytree D. For an algorithm output A and an optimal polytree $S^\star$, Theorem 3.6 guarantees that $s_{\mathrm{shift}}(S^\star) \le \rho\, s_{\mathrm{shift}}(A)$, which, after substituting the definition of s_{shift} on both sides and rearranging, becomes

$$s_{\mathrm{orig}}(S^\star) \;\le\; \rho\, s_{\mathrm{orig}}(A) + (\rho - 1)\, C. \tag{45}$$

The BDeu scores are negative, so $|s_{\mathrm{orig}}(D)| = -s_{\mathrm{orig}}(D)$ for every polytree D. Multiplying (45) by -1 gives

$$|s_{\mathrm{orig}}(S^\star)| \;\ge\; \rho\,|s_{\mathrm{orig}}(A)| - (\rho - 1)\, C,$$

equivalently $\rho\,|s_{\mathrm{orig}}(A)| \le |s_{\mathrm{orig}}(S^\star)| + (\rho - 1)\, C$, and dividing by ρ and then by $|s_{\mathrm{orig}}(S^\star)|$ yields the multiplicative bound

$$\frac{|s_{\mathrm{orig}}(A)|}{|s_{\mathrm{orig}}(S^\star)|} \;\le\; \frac{1}{\rho} + \left(1 - \tfrac{1}{\rho}\right) \frac{C}{|s_{\mathrm{orig}}(S^\star)|}. \tag{46}$$

The right-hand side still depends on the unknown $|s_{\mathrm{orig}}(S^\star)|$. To bound it from above by a data-computable quantity, observe that the per-node maxima ignore acyclicity and therefore

$$s_{\mathrm{orig}}(S^\star) \;\le\; \textstyle\sum_v \max_P f_{v,\mathrm{orig}}(P);$$

on each of our datasets the right-hand side is negative, so taking absolute values reverses the inequality and gives

$$|s_{\mathrm{orig}}(S^\star)| \;\ge\; \left|\textstyle\sum_v \max_P f_{v,\mathrm{orig}}(P)\right|.$$

Substituting this lower bound into (46) yields the data-computable conservative factor

$$\frac{|s_{\mathrm{orig}}(A)|}{|s_{\mathrm{orig}}(S^\star)|} \;\le\; \underbrace{\frac{1}{\rho} + \left(1 - \tfrac{1}{\rho}\right) \frac{C}{\left|\sum_v \max_P f_{v,\mathrm{orig}}(P)\right|}}_{=:\,\rho_{\mathrm{orig}}}. \tag{47}$$